

Guide to the Systems Engineering Body of Knowledge (SEBoK) version 1.9.1

16 OCTOBER 2018

Guide to the Systems Engineering Body of Knowledge (SEBoK)

Version 1.9.1

PDF generated using the open source mwlib toolkit. See http://code.pediapress.com/ for more information.
PDF generated at: Tue, 16 Oct 2018 07:58:47 CEST

Contents

Articles

Combined Examples

References

Letter from the Editor



A very warm welcome to all SEBoK users!

The BKCASE Editor in Chief (EIC) has overall responsibility for the continuing review and update of the SEBoK. Many thanks to the BKCASE Governors and the current members of the Editorial Board for supporting me.

It is an honor to become the third EIC for the SEBoK. Since being named to this position, I have held listening sessions at the INCOSE IS in Washington, D.C., met with the Board of Governors and the SEBoK Editors for the various parts of the SEBoK. Let me share some of those ideas and insights with you. Below are the more significant inputs:

1. A large number of systems engineers know of the SEBoK, and have used it from time to time
2. The PDF version of the SEBoK is very popular - it is portable and searchable - and we should continue to produce this product with each new release of the SEBoK
3. The organization of the SEBoK still needs some tweaking
4. There are recommendations for potential new sections to be added: SE Research, SE 2025 Vision, and Systems Engineering for Non-systems engineers
5. Additional topics that should be added: Systems Thinking, TRIZ, Digital Thread/Digital Twin are some of the recommendations
6. Provide more related engineering topics: Civil engineering, Coastal engineering, Mechanical engineering, Data engineering, etc.
7. Need a better tool for submitting new knowledge to the SEBoK editors
8. Time to incorporate multimedia into the SEBoK
9. Add more examples of successful systems implementations, tool information, and modeling techniques

From those listening opportunities, a vision for SEBoK 2.0 has emerged.

SEBoK 2.0 will be multimedia enabled. We will begin to incorporate video and audio links. Because we need to be sensitive to copyrights, original content from our user base will be solicited. If you are an expert in areas that are already a part of the SEBoK, and would like to create short, 3-5 minute vignettes, please contact one of the editors or myself. We will also be leveraging the INCOSE YouTube channel for some of this content. Since many are accessing the SEBoK via their phones and tablets, we will work to improve the accessibility on mobile devices. Finally, we will continue to fine-tune the organization of the SEBoK, adding more content "as appropriate".

Why put "as appropriate" in quotes? As you may know, the SEBoK is a curated Wiki. That means that the information contained in the SEBoK should be established knowledge. While research is becoming more and more important to systems engineering, it is also important to differentiate what is new knowledge, and what is considered 'settled' knowledge, or general knowledge. To handle this, the editors are considering adding a new section to the SEBoK where research type topics can be cataloged, but with the understanding that it is still research.

The SEBoK Editorial Board is always seeking more collaborators and new and/or updated content. It is important to note that the editors primary role is not to create the content (though many of them are also content creators). It is to edit and curate the content. Content creation should be coming from the consumers of the SEBoK.



SEBoK Version 1.9.1 Changes ========================= This minor update, Version 1.9.1 contains minor updates - fixed links, updated Editorial Board members and their contact information, and this letter.

BKCASE Governance and Editorial Board

BKCASE Governing Board

The three SEBoK steward organizations – the International Council on Systems Engineering (INCOSE), the Institute of Electrical and Electronics Engineers Computer Society (IEEE-CS), and the Systems Engineering Research Center (SERC) provide the funding and resources needed to sustain and evolve the SEBoK and make it available as a free and open resource to all. The stewards appoint the BKCASE Governing Board to be their primary agents to oversee and guide the SEBoK and its companion BKCASE product, GRCSE.

The BKCASE Governing Board includes:

- **The International Council on Systems Engineering (INCOSE)**
  - Art Pyster (Governing Board Chair), Paul Frenz
- **Systems Engineering Research Center (SERC)**
  - Jon Wade, Cihan Dagli
- **IEEE Computer Society (IEEE CS)**
  - Andy Chen, Rich Hilliard

Past INCOSE governors Bill Miller, Kevin Forsberg, David Newbern, David Walden, Courtney Wright, Dave Olwell, Ken Nidiffer, Richard Fairley, Massood Towhidnejad, and John Keppler. The governors would also like to acknowledge John Keppler, IEEE Computer Society, who has been instrumental in helping the Governors to work within the IEEE CS structure.

The stewards appoint the BKCASE Editor in Chief to manage the SEBoK and GRCSE and oversee the Editorial Board.

Editorial Board

The SEBoK Editorial Board is chaired by an Editor in Chief, supported by a group of Associate Editors.

BKCASE Editor in Chief



Robert J. Cloutier

University of South Alabama

rcloutier@southalabama.edu [1]

Responsible for the appointment of SEBoK Editors and for the overall content and coherence of the SEBoK.

Each Editor has his/her area(s) of responsibility, or shared responsibility, highlighted in the table below.

SEBoK Part 1 SEBoK Introduction

Ariela Sofer

George Mason University (USA)

asofer@gmu.edu [2]

Responsible for Part 1

SEBoK Part 2: Systems

Cihan Dagli

Missouri University of Science & Technology (USA)

dagli@mst.edu [3]

Responsible for the Systems Approach Applied to Engineered Systems knowledge areas

Dov Dori

Massachusetts Institute of Technology (USA) and Technion Israel Institute of Technology (Israel)

dori@mit.edu [4]

Responsible for the Representing Systems with Models knowledge area

Duane Hybertson

MITRE (USA)

dhyberts@mitre.org [5]

Jointly responsible for the Systems Fundamentals, Systems Science and Systems Thinking knowledge areas

Janet Singer

UC Santa Cruz (USA)

jsinger@soe.ucsc.edu [6]

Jointly responsible for the Systems Fundamentals, Systems Science and Systems Thinking knowledge areas

Peter Tuddenham

College of Exploration (USA)

Peter@coexploration.net [7]

SEBoK Part 3: Systems Engineering and Management

Barry Boehm

University of Southern California (USA)

boehm@usc.edu [8]

Jointly responsible for the Systems Engineering Management and Life Cycle Models knowledge areas

Kevin Forsberg

OGR Systems

kforsberg@ogrsystems.com [9]

Jointly responsible for the Systems Engineering Management and Life Cycle Models knowledge areas

Gregory Parnell

University of Arkansas (USA)

gparnell@uark.edu [10]

Responsible for Systems Engineering Management knowledge area.

Garry Roedler

Lockheed Martin (USA)

garry.j.roedler@lmco.com [11]

Responsible for the Concept Definition and System Definition knowledge areas.

Phyllis Marbach

Incose LA (USA)

prmarbach@gmail.com [12]

Ken Zemrowski

ENGILITY

kenneth.zemrowski@incose.org [13]

Responsible for the Systems Engineering Standards knowledge area.

SEBoK Part 4: Applications of Systems Engineering

Judith Dahmann

MITRE Corporation (USA)

jdahmann@mitre.org [14]

Jointly responsible for Product Systems Engineering and Systems of Systems (SoS) knowledge areas

Michael Henshaw

Loughborough University (UK)

M.J.d.Henshaw@lboro.ac.uk [15]

Jointly responsible for Product Systems Engineering and Systems of Systems (SoS) knowledge areas

James Martin

The Aerospace Corporation

james.martin@incose.org [16]

Responsible for the Enterprise Systems Engineering knowledge area.

SEBoK Part 5: Enabling Systems Engineering

Emma Sparks

Cranfield University

Jointly responsible for the Enabling IndividualsandEnabling Teams knowledge area

Rick Hefner

California Institute of Technology

Rick.Hefner@ngc.com [17]

Tim Ferris *Cranfield University*

Timothy.Ferris@cranfield.ac.uk [18]

Bernardo Delicado

MBDA / INCOSE

bernardo.delicado@mbda-systems.com [19]

SEBoK Part 6 Related Disciplines

Alice Squires

Washington State University (USA)

alice.squires@wsu.edu [20]

Paul Phister

MANIAC Consulting

phisterp@juno.com [21]

SEBoK Part 7 Systems Engineering Implementation Examples

Dick Fairley

Software & Systems Engineering Associates (USA)

[mailto: dickfairley@gmail.com dickfairley@gmail.com]

Jointly responsible for Part 7: Systems Engineering Implementation Examples, which includes Case Studies and examples.

Clif Baldwin

FAA Technical Center

cliftonbaldwin@gmail.com [22]

Jointly responsible for Part 7: Systems Engineering Implementation Examples, which includes Case Studies and examples.

Graduate Reference Curriculum for Systems Engineering (GRCSE)

David H. Olwell

St. Martin's University (USA)

dolwell@stmartin.edu [23]

Associate Editor for SEBoK.

Editorial Board Support

The Assistant Editor provide general editorial support across all topics and assist with both content improvement and production issues.

BKCASE Technical Director

Nicole Hutchison

Stevens Institute of Technology (USA)

bkcase.incose.ieeecs@gmail.com [24]

Interested in Editing?

The Editor in Chief is looking for additional editors to support the evolution of the SEBoK. Editors are responsible for maintaining and updating one to two knowledge areas, including recruiting and working with authors, ensuring the incorporation of community feedback, and maintaining the quality of SEBoK content. We are specifically interested in support for the following knowledge areas:

- System Deployment and Use
- Product and Service Life Management
- Enabling Businesses and Enterprises
- Systems Engineering and Software Engineering
- Procurement and Acquisition
- Systems Engineering and Specialty Engineering

If you are interested in being considered for participation on the Editorial Board, please visit the BKCASE website http://www.bkcase.org/join-us/or contact the BKCASE Staff directly at bkcase.incose.ieeecs@gmail.com [24].

SEBoK v. 1.9.1, released 16 October 2018

References

[1] mailto:rcloutier@southalabama.edu
[2] mailto:asofer@gmu.edu
[3] mailto:dagli@mst.edu
[4] mailto:dori@mit.edu
[5] mailto:dhyberts@mitre.org
[6] mailto:jsinger@soe.ucsc.edu
[7] mailto:Peter@coexploration.net
[8] mailto:boehm@usc.edu
[9] mailto:kforsberg@ogrsystems.com
[10] mailto:gparnell@uark.edu
[11] mailto:garry.j.roedler@lmco.com
[12] mailto:prmarbach@gmail.com
[13] mailto:kenneth.zemrowski@incose.org
[14] mailto:jdahmann@mitre.org
[15] mailto:M.J.d.Henshaw@lboro.ac.uk
[16] mailto:james.martin@incose.org
[17] mailto:Rick.Hefner@ngc.com
[18] mailto:Timothy.Ferris@cranfield.ac.uk
[19] mailto:bernardo.delicado@mbda-systems.com
[20] mailto:alice.squires@wsu.edu
[21] mailto:phisterp@juno.com
[22] mailto:cliftonbaldwin@gmail.com
[23] mailto:dolwell@stmartin.edu
[24] mailto:bkcase.incose.ieeecs@gmail.com

Acknowledgements and Release History

This article describes the contributors to the current version of the SEBoK. For information on contributors to past versions of the SEBoK, please follow the links under "SEBoK Release History" below. To learn more about the updates to the SEBoK for v. 1.9.1, please see the Letter from the Editor.

The BKCASE Project began in the fall of 2009. Its aim was to add to the professional practice of systems engineering by creating two closely related products:

- *Guide to the Systems Engineering Body of Knowledge (SEBoK)*
- *Graduate Reference Curriculum for Systems Engineering (GRCSE)*

BKCASE History, Motivation, and Value

The **Guide to the Systems Engineering Body of Knowledge (SEBoK)** is a living authoritative guide that discusses knowledge relevant to Systems Engineering. It defines how that knowledge should be structured to facilitate understanding, and what reference sources are the most important to the discipline. The curriculum guidance in the **Graduate Reference Curriculum for Systems Engineering (GRCSE)** (Pyster and Olwell et al. 2015) makes reference to sections of the SEBoK to define its core knowledge; it also suggests broader program outcomes and objectives which reflect aspects of the professional practice of systems engineering as discussed across the SEBoK.

Between 2009 and 2012 BKCASE was led by Stevens Institute of Technology and the Naval Postgraduate School in coordination with several professional societies and sponsored by the U.S. Department of Defense (DoD), which provided generous funding. More than 75 authors and many other reviewers and supporters from dozens of companies, universities, and professional societies across 10 countries contributed many thousands of hours writing the SEBoK articles; their organizations provided significant other contributions in-kind.

The SEBoK came into being through recognition that the systems engineering discipline could benefit greatly by having a living authoritative guide closely related to those groups developing guidance on advancing the practice, education, research, work force development, professional certification, standards, etc.

At the beginning of 2013, BKCASE transitioned to a new governance model with shared stewardship between the Systems Engineering Research Center (SERC) [1], the International Council on Systems Engineering (INCOSE) [2], and the Institute of Electrical and Electronics Engineers Computer Society (IEEE-CS) [3]. This governance structure was formalized in a memorandum of understanding between the three stewards that was finalized in spring of 2013. The stewards have reconfirmed their commitment to making the SEBoK available at no cost to all users, a key principle of BKCASE.

Please see http://www.bkcase.org for more information or signup for the BKCASE newsletter [4].

As of the end of February 2016, SEBoK articles have been accessed more than 1,000,000 times. We hope the SEBoK will regularly be used by thousands of systems engineers and others around the world as they undertake technical activities such as eliciting requirements, creating systems architectures, or analysis system test results; and professional development activities such as developing career paths for systems engineers, deciding new curricula for systems engineering university programs, etc.

Governance

The SEBoK is shaped by the BKCASE Editorial Board and is overseen by the BKCASE Governing Board. A complete list of members for each of these bodies can be found on the BKCASE Governance and Editorial Board page.

Content and Feature Updates for 1.9.1

This version of the SEBoK was released 12th October 2018. This is a micro release of the SEBoK which includes the new Editor in Chief, R.J. Cloutier, updates to the editorial board, and a number of updates to the wiki software. Software updates include the newest version of MediaWiki, a switch to the new comment feature of MediaWiki, replacing the previous comment feature of DISQUS. The SEBoK PDF was also updated (see Download SEBoK PDF).

For more information about this release please refer to Version 1.9.1.

SEBoK Release History

There have been 19 releases of the SEBoK to date, collected into 10 main releases.

Main Releases

- Version 1.9.1 - current version.
- Version 1.9 - A minor update which included updates to the System Resilience article in Part 6: Related Disciplines, as well as a major restructuring of Part 7: Systems Engineering Implementation Examples. A new example has been added around the use of model based systems engineering for the thirty-meter telescope.
- Version 1.8 - A minor update, including an update of the Systems of Systems (SoS) knowledge area in Part 4: Applications of Systems Engineering where a number of articles were updated on the basis of developments in the area as well as on comments from the SoS and SE community. Part 6: Related Disciplines included updates to the Manufacturability and Producibility and Reliability, Availability, and Maintainability articles.
- Version 1.7 - A minor update, including a new Healthcare SE Knowledge Area (KA), expansion of the MBSE area with two new articles, Technical Leadership and Reliability, Availability, and Maintainability and a new case study on the Northwest Hydro System.

- Version 1.6 - A minor update, including a reorganization of Part 1 SEBoK Introduction, a new article on the Transition towards Model Based Systems Engineering and a new article giving an overview of Healthcare Systems Engineering, a restructure of the Systems Engineering and Specialty Engineering KA.
- Version 1.5 - A minor update, including a restructure and extension of the Software Engineering Knowledge Area, two new case studies, and a number of corrections of typographical errors and updates of outdated references throughout the SEBoK.
- Version 1.4 - A minor update, including changes related to ISO/IEC/IEEE 15288:2015 standard, three new case studies and updates to a number of articles.
- Version 1.3 - A minor update, including three new case studies, a new use case, updates to several existing articles, and updates to references.
- Version 1.2 - A minor update, including two new articles and revision of several existing articles.
- Version 1.1 - A minor update that made modest content improvements.
- Version 1.0 - The first version intended for broad use.

Click on the links above to read more information about each release.

Wiki Team

In January 2011, the authors agreed to move from a document-based SEBoK to a wiki-based SEBoK, and beginning with v. 0.5, the SEBoK has been available at www.sebokwiki.org [5] Making the transition to a wiki provided three benefits:

1. easy worldwide access to the SEBoK;
2. more methods for search and navigation; and
3. a forum for community feedback alongside content that remains stable between versions.

The wiki team is responsible for maintenance of the wiki infrastructure as well as technical review of all materials prior to publication.

- Nicole Hutchison, Stevens Institute of Technology.

The wiki is currently supported by Ike Hecht from WikiWorks.

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.sercuarc.org
[2] http://www.incose.org
[3] http://www.computer.org
[4] http://www.bkcase.org/subscribe/
[5] http://www.sebokwiki.org

Part 1: SEBoK Introduction

SEBoK Introduction

The purpose of the *Guide to the Systems Engineering Body of Knowledge (SEBoK)* is to provide a widely accepted, community-based, and regularly updated baseline of Systems Engineering (glossary) (SE) knowledge. SEBoK Part 1 contains an introduction to both the discipline of SE and an introduction to the SEBoK wiki and how to use it.

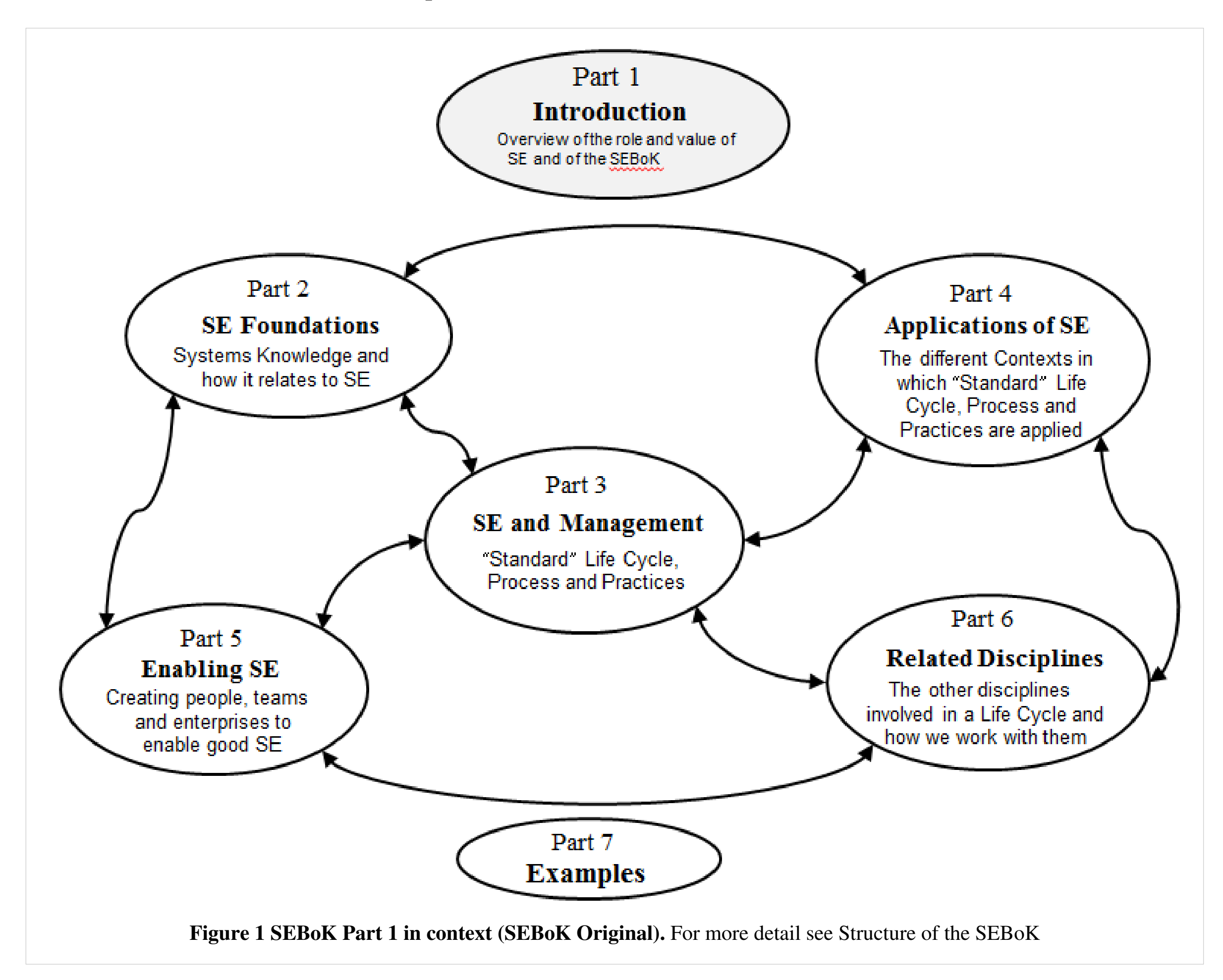


Figure 1 SEBoK Part 1 in context (SEBoK Original). For more detail see Structure of the SEBoK

Part 1 also includes an introduction to some of the emerging aspects of systems engineering and a discussion of how these are transforming the discipline. As this knowledge matures it will be migrated into the main body of the SEBoK.

Part 1 Knowledge Areas

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 1 contains the following KAs:

- Introduction to the SEBoK
- Introduction to Systems Engineering
- Introduction to SE Transformation
- SEBoK Users and Uses

Scope and Context of the SEBoK

The SEBoK is one of two complementary products. The other, which uses the content of the SEBoK to define a core body of knowledge (CorBoK) to be included in graduate SE curricula, is called the Graduate Reference Curriculum for Systems Engineering (GRCSE™). GRCSE is *not* a standard, but a reference curriculum to be tailored and extended to meet the objectives of each university's graduate program. (Pyster and Olwell et al. 2015) These products are being developed by the Body of Knowledge and Curriculum to Advance Systems Engineering (BKCASE) [1] project.

Most of the SEBoK (Parts 2 – 6) focuses on domain-independent information—that which is universal to systems engineering regardless of the domain in which it is applied. Part 7 includes examples from real projects. These illustrate the concepts discussed elsewhere in the SEBoK, while detailing considerations relevant to domains such as aerospace, medicine, and transportation.

SE in the context of engineered systems (ES) is the primary scope for the SEBoK, though general systems concepts are also discussed in Part 2. The SEBoK also covers considerations for the disciplines of software engineering and project management, which are strongly intertwined with the practice of SE (see Part 6).

References

Works Cited

Pyster, A., D.H. Olwell, T.L.J. Ferris, N. Hutchison, S. Enck, J.F. Anthony, D. Henry, and A. Squires (eds). 2015.*Graduate Reference Curriculum for Systems Engineering (GRCSE),*version 1.1. Hoboken, NJ, USA: The Trustees of the Stevens Institute of Technology ©2015. Accessed on 4 December 2015 at BKCASE.orghttp://www.bkcase.org/grcse-2/.

Primary References

Pyster, A., D.H. Olwell, T.L.J. Ferris, N. Hutchison, S. Enck, J.F. Anthony, D. Henry, and A. Squires (eds). 2015.*Graduate Reference Curriculum for Systems Engineering (GRCSE™),* version 1.1. Hoboken, NJ, USA: The Trustees of the Stevens Institute of Technology ©2015. Accessed on 4 December 2015 at BKCASE.orghttp://www.bkcase.org/grcse-2/.

< Return to Table of Contents | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.bkcase.org/

Knowledge Area: Introduction to the SEBoK

Introduction to the SEBoK

The SEBoK provides a widely accepted, community-based, and regularly updated baseline of Systems Engineering (glossary) (SE) knowledge. This baseline will strengthen the mutual understanding across the many disciplines involved in developing and operating systems.

Topics

Each part of the SEBoK is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Scope of the SEBoK
- Structure of the SEBoK

References

Works Cited

none

Primary References

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.

Sage, A. and W. Rouse (eds). 2009. *Handbook of Systems Engineering and Management,* 2nd ed. Hoboken, NJ, USA: John Wiley and Sons, Inc.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Scope of the SEBoK

The SEBoK is a large compendium of information about systems engineering. It:

- is a guide to the body of SE knowledge which provides references to detailed sources for additional information; it is not a self-contained knowledge resource
- focuses on Engineered Systems contexts, that is socio-technical systems with a recognized SE life cycle,
- while treating social and natural systems as relevant and important environmental considerations (see Part 2)
- describes generic SE life cycle and process knowledge (see Part 3)
- recognizes that SE principles can be applied differently to different types of products, services, enterprises, and systems of systems (SoS) context (see Part 4)
- provides resources for organization support of SE activities (see Part 5)
- explores the interaction between SE and other disciplines, highlighting what systems engineers need to know about these disciplines (see Part 6)
- is domain-independent, with implementation examples to provide domain-specific context (see Part 7)

Each of these considerations depends upon the definition and scope of SE itself, which is the subject of the next section.

SEBoK Purposes

Ongoing studies of system cost and schedule failures (Gruhl & Stutzke 2005; Johnson 2006) and safety failures (Leveson 2012) have shown that the failures have mostly come not from their domain disciplines, but from lack of adequate SE. To provide a foundation for the mutual understanding of SE needed to reduce these failure, the SEBoK describes the boundaries, terminology, content, and structure of SE. In so doing, the SEBoK systematically and consistently supports six broad purposes, described in Table 1.

Table 1. SEBoK Purposes. (SEBoK Original)

#	Purpose	Description
1	Inform Practice	Inform systems engineers about the boundaries, terminology, and structure of their discipline and point them to useful information needed to practice SE in any application domain.
2	Inform Research	Inform researchers about the limitations and gaps in current SE knowledge that should help guide their research agenda.
3	Inform Interactors	Inform performers in interacting disciplines (system implementation, project and enterprise management, other disciplines) and other stakeholders of the nature and value of SE.
4	Inform Curriculum Developers	Inform organizations defining the content that should be common in undergraduate and graduate programs in SE.
5	Inform Certifiers	Inform organizations certifying individuals as qualified to practice systems engineering.
6	Inform SE Staffing	Inform organizations and managers deciding which competencies that practicing systems engineers should possess in various roles ranging from apprentice to expert.

The SEBoK is a guide to the body of SE knowledge, not an attempt to capture that knowledge directly. It provides references to more detailed sources of knowledge, all of which are generally available to any interested reader. No proprietary information is referenced, but not all referenced material is free—for example, some books or standards must be purchased from their publishers. The criterion for including a source is simply that the authors believed it offered the best generally available information on a particular subject.

The SEBoK is global in applicability. Although SE is practiced differently from industry to industry and country to country, the SEBoK is written to be useful to systems engineers anywhere. The authors were chosen from diverse

locales and industries, and have refined the SEBoK to broaden applicability based on extensive global reviews of several drafts.

The SEBoK aims to inform a wide variety of user communities about essential SE concepts and practices, in ways that can be tailored to different enterprises and activities while retaining greater commonality and consistency than would be possible without the SEBoK. Because the world in which SE is being applied is evolving and dynamic, the SEBoK is designed for easy, continuous updating as new sources of knowledge emerge.

SEBoK Uses

The communities involved with SE include its various specialists, engineers from disciplines other than systems engineering, managers, researchers, and educators. This diversity means that there is no single best way to use the SEBoK. The SEBoK includes use cases that highlight ways that particular communities can draw upon the content of the SEBoK, identify articles of interest to those communities, and discuss primary users (those who use the SEBoK directly), and secondary users (those who use the SEBoK with assistance from a systems engineer). See the article SEBoK Users and Uses.

SEBoK Domain Independent Context

The SEBoK uses language and concepts that are generally accepted for domain-independent SE. For example, the domain-independent conceptual foundations of SE are elaborated in Part 2: Foundations of Systems Engineering. However, each of the numerous domains in which SE is practiced — including telecommunications, finance, medicine, and aerospace — has its own specialized vocabulary and key concepts. Accordingly, the SEBoK is designed to show how its domain-independent material relates to individual domains in two ways.

Firstly, by means of examples that tell stories of how SE is applied in particular domains. Part 7: Systems Engineering Implementation Examples) consists of examples (case studies and vignettes), each set in a particular domain such as aerospace, medicine, or software, and featuring vocabulary and concepts special to that domain. There are similar vignettes in some of the Use Cases in Part 1. These examples demonstrate the effect of domain on the application of SE and complement the domain-independent information elsewhere in the SEBoK. They show how a concept works in a given domain and provide a fair opportunity for reviewers to reflect on whether there are better ways to capture application-dependent aspects of SE knowledge.

In addition, the SEBoK will contain knowledge areas in Part 4: Applications of Systems Engineering which explicitly describe the domain specific language, approaches, specialized processes and tools, etc. of particular application domains. In this version of the SEBoK there are a limited set of domain knowledge areas.

The SEBoK authors recognise the value of both case studies and domain extensions, both will be expanded in later versions.

References

Works Cited

Gruhl, W. and Stutzke, R. 2005. "Werner Gruhl Analysis of SE Investments and NASA Overruns," in R. Stutzke, *Estimating Software-Intensive Systems*. Boston, MA, USA: Addison Wesley, page 290.

Johnson, J. 2006. *My Life Is Failure: 100 Things You Should Know to Be a Better Project Leader*. Boston, MA, USA: Standish Group International.

Leveson, N. 2012. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA, USA: MIT Press.

Primary References

none.

Additional References

none

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Structure of the SEBoK

The **Guide to the Systems Engineering Body of Knowledge (SEBoK)** is a living authoritative guide that discusses knowledge relevant to Systems Engineering. SEBoK does not contain all of this knowledge itself, but provides a starting point and key resources to allow the reader to navigate the wider body of knowledge that exists in published sources. To do this SEBoK:

- Defines how relevant knowledge should be structured to facilitate understanding.
- Provides short discussions of key idea, principles and concepts within that structure.
- Points to reference sources important to the discipline, which explore these ideas in more detail.

In doing this it is inevitable that differences in terminology, alternative approaches and even fundamentally different ways of thinking within the knowledge will appear. SEBoK attempts were possible to provide clarity of similar or overlapping idea, or to highlight real differences and the reasons behind them. In particular the SEBoK Glossary of Terms contains the most used or generally agreed definitions of terms when it can, but may highlight more than one definition if needed to show breadth of current thinking.

SEBoK Structure

Figure 1, below, gives a summary of the 7 parts of the SEBoK and how they are related.

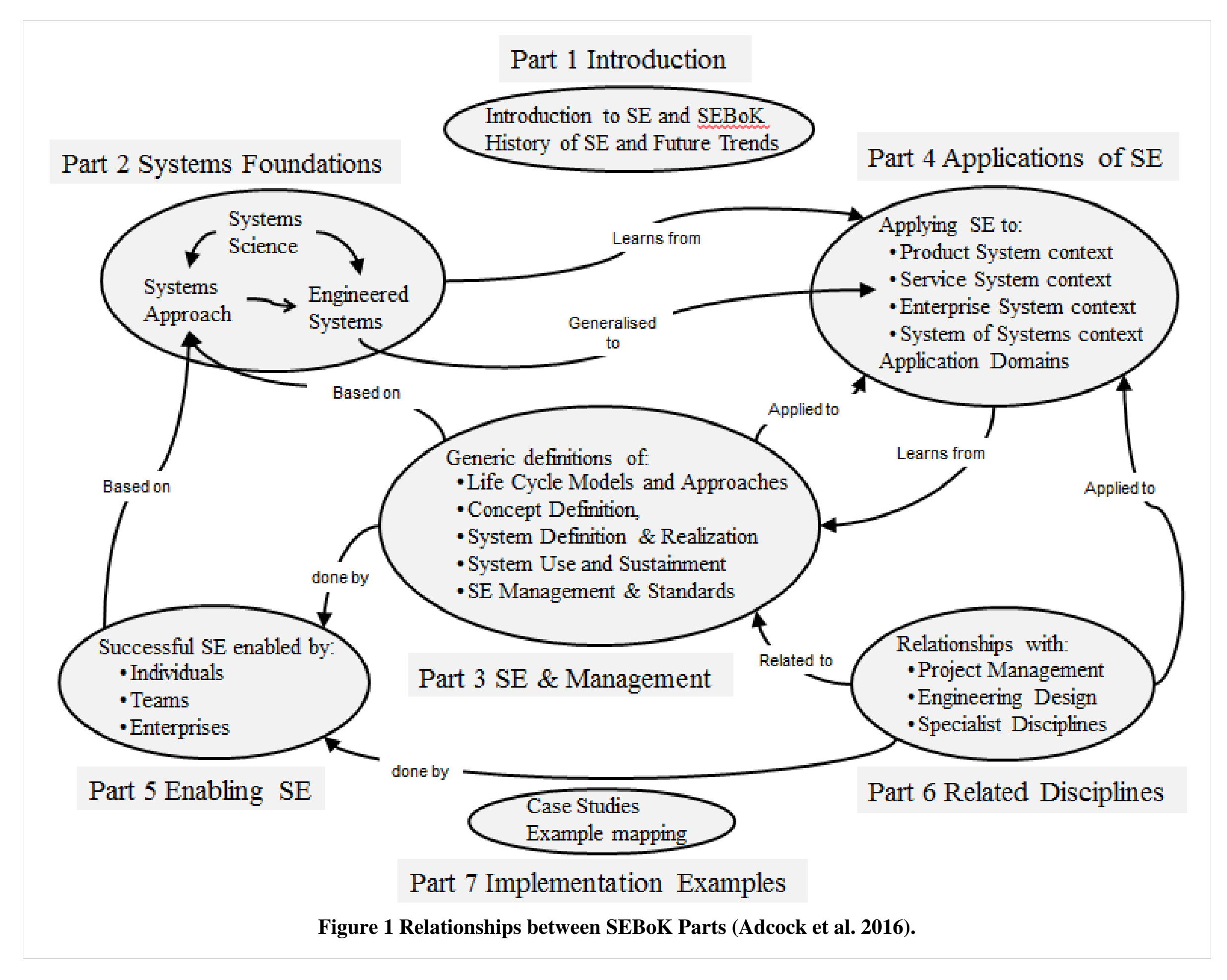


Figure 1 Relationships between SEBoK Parts (Adcock et al. 2016).

The scope of each part and the key relationships amongst them is briefly discussed below. For a more detailed discussion of how this structure was evolved see (Adcock et al, 2016).

Overview of Parts

Part 1: SEBoK Introduction

This part explains the scope, context, and structure of the SEBoK, and of systems engineering (SE).

An overview of who should use the SEBoK, and for what purpose, is followed by detailed use cases. The economic value, history, and relationship to other disciplines are discussed. Part 1 also contains a section which discussed the future evolution of the SEBoK and allows for new areas of content to be introduced before being transitioned into other SEBoK parts.

Part 2: Foundations of Systems Engineering

This part provides an introduction and overview of areas of knowledge which provide the foundations of SE.

A discussion of the definitions and basic concepts of System (glossary) is followed by an overview of the principles, concepts, methods, models and patterns of some of the key foundational areas of Systems Science (glossary). This includes a detailed consideration of the foundational knowledge related to systems models and modelling.

Part 2 looks in more detail at two aspects of this foundational knowledge of particular value to SE. The first is to discuss aspects of systems knowledge related to a Systems Approach (glossary) to complex problems and opportunities. This approach provides foundations for how SE is defined and practices (see Parts 3 and 5 below). The second is to describe the different ways in which system concepts are applied to real world concerns. The SEBoK defines an engineered system (ES) as the primary focus for the application of SE (see Part 4 below).

Part 3: Systems Engineering and Management

This part describes generic knowledge on the practice of SE and related management activities.

Part 3 begins with the life cycle models common in SE and the general principles behind their application. It then moves on to SE management activities. Covering both technical activities such as requirements, architecture, test and evaluation; and management activities such as planning, measurement, risk. Next is product and service life management, a distinct area of SE management that emphasizes the entire life cycle including retirement and disposal. An account of SE standards concludes this part.

Focused on what many think of as the main body of SE, including best practices and common pitfalls, this part constitutes a substantial proportion of the SEBoK. As already discussed, the knowledge in Part 3 is based on the systems approach from Part 2. The links between Part 3 and the other parts of the SEBoK are discussed below. It is anticipated that this part and the following parts will reflect increased emphasis on model-based systems engineering (MBSE) practices as these practices continue to evolve and become more mainstream.

Part 4: Applications of Systems Engineering

This part describes how to apply SE principles to different types of System Context (glossary).

Part 4 focuses on four major Engineered System (glossary) contexts in turn: products, services, enterprises, and systems of systems (SoS). For each one the system abstraction, commercial relationships and application of generic SE is described.

The generalized contexts above should be viewed as overlapping models of how SE can be applied in different kinds of situation. Combinations of one or more of them are fully realized when applied in an application domain. Part 4 currently described this application in a small number of such domains. This will be expanded in later updates. The applications of SE in this part describe the real world practice of SE. The generalized knowledge in both Parts 2 and 3 evolves through what we learn from these applications. Part 2 includes a discussion of this relationship between theory and practice.

Part 5: Enabling Systems Engineering

This part describes how to organize to enable the success performance of SE activities.

Part 4 covers knowledge at the enterprise, team, or individual level. The range of considerations extends from value proposition, business purpose, and governance, down to competency, personal development as a systems engineer, and ethics.

All of these relate to the baseline definitions of SE in Part 3, further generalized in the levels of application in Part 4. The systems approach in Part 2 should also form a foundation for this part. Since the practice of SE is multi-disciplinary, Part 5 also has a link to Part 6 as discussed below.

Part 6: Related Disciplines

This part describes the relationships between SE and other disciplines.

Part 6 covers the links between SE and software engineering (SwE), project management (PM), industrial engineering and procurement. It also describes how SE is related to specialty engineering, which describes the various system "–ilities" (like reliability, availability, and maintainability) that SE must balance and integrate.

The knowledge in this part provides an interface to other bodies of knowledge, focused on how it is linked to Parts 3, 5 and 5 above.

Part 7: Systems Engineering Implementation Examples

A set of real-world examples of SE activities forms the natural conclusion of the SEBoK. These come in two forms: case studies, which refer the reader to and summarize published examinations of the successes and challenges of SE programs, and vignettes, which are brief, self-contained wiki articles. This part is a key place to look within the SEBoK for lessons learned, best practices, and patterns. Many links connect material in the examples to the conceptual, methodological, and other content elsewhere in the SEBoK.

Addenda

The SEBoK contains a Glossary of Terms, which provides authoritatively-referenced definitions of key terms. It also contains a list of Primary References, with additional information about each reference. Quicklinks in the left margin provide additional background information, including a table of contents, a listing of articles by topic [1], and a list of Acronyms.

References

Works Cited

Adcock, R., Hutchison, N., Nielsen, C., 2016, "Defining an architecture for the Systems Engineering Body of Knowledge," Annual IEEE Systems Conference (SysCon) 2016.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://sebokwiki.org/1.1.1/index.php?title=Category:Topic

Knowledge Area: Introduction to Systems Engineering

Introduction to Systems Engineering

The primary focus of the SEBoK is on the current baseline of knowledge describing the practice of domain independent systems engineering (SE). This Knowledge Area (KA) contains topic articles which provide an overview of SE practice and discuss its economic value, historic evolution and key relationships.

Topics

Each part of the SEBoK is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Systems Engineering Overview
- Economic Value of Systems Engineering
- Systems Engineering: Historic and Future Challenges
- Systems Engineering and Other Disciplines

Systems Engineering

SE is an interdisciplinary approach and means to enable the realization of successful systems. Successful systems must satisfy the needs of its customers, users and other stakeholders. Some key elements of systems engineering are highlighted in Figure 1 and include:

- The principles and concepts that characterize a system, where a system is an interacting combination of system elements to accomplish a defined objective(s). The system interacts with its environment, which may include other systems, users, and the natural environment. The system elements that compose the system may include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements.
- A systems engineer is a person or role who supports this interdisciplinary approach. In particular, the systems engineer often serves to elicit and translate customer needs into specifications that can be realized by the system development team.
- In order to help realize successful systems, the systems engineer supports a set of life cycle processes beginning early in conceptual design and continuing throughout the life cycle of the system through its manufacture, deployment, use and disposal. The systems engineer must analyze, specify, design, and verify the system to ensure that its functional, interface, performance, physical, and other quality characteristics, and cost are balanced to meet the needs of the system stakeholders.
- A systems engineer helps ensure the elements of the system fit together to accomplish the objectives of the whole, and ultimately satisfy the needs of the customers and other stakeholders who will acquire and use the system.

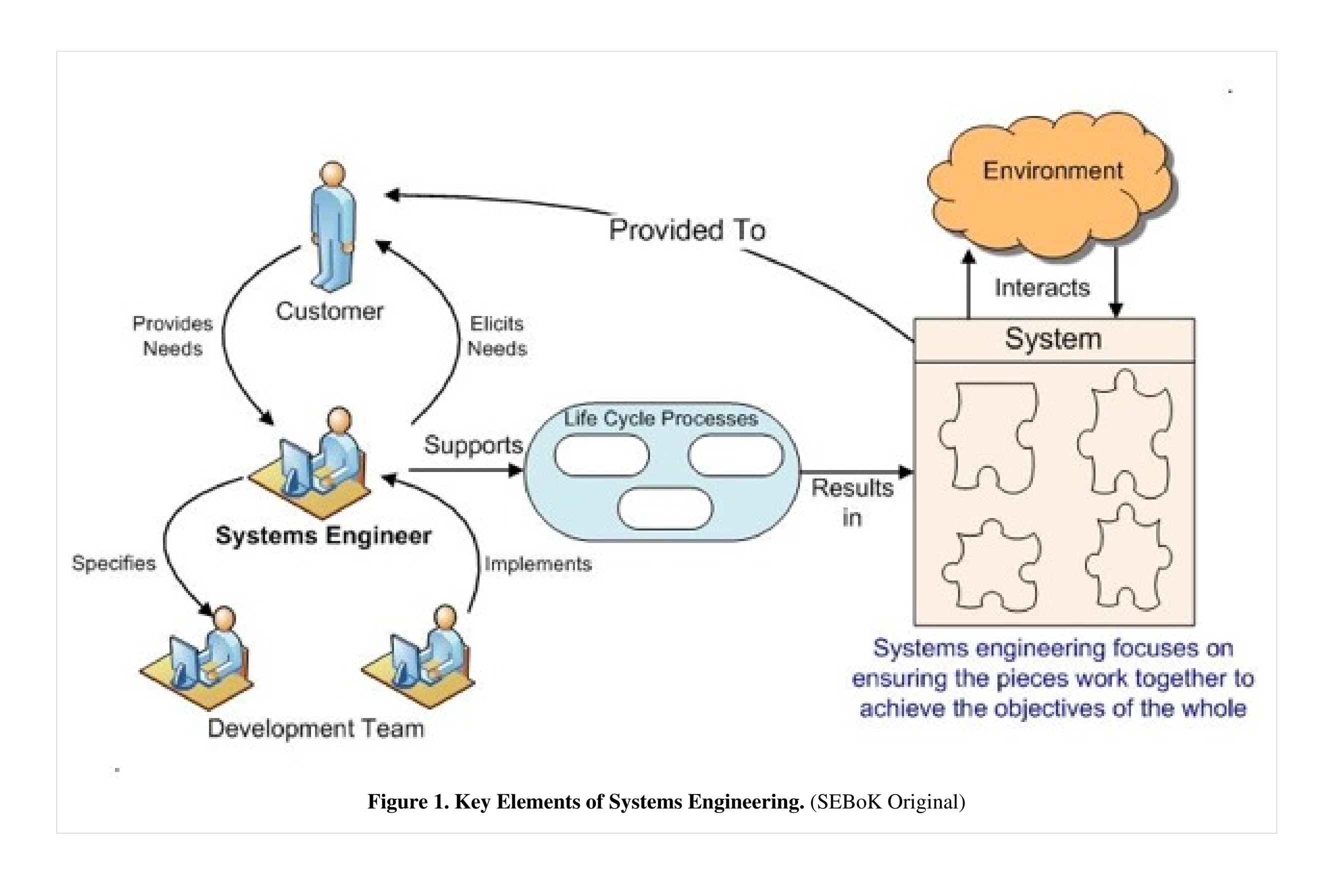


Figure 1. Key Elements of Systems Engineering. (SEBoK Original)

References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Systems Engineering Overview

Systems engineering (SE) is an interdisciplinary approach and means to enable the realization of successful systems. Successful systems must satisfy the needs of their customers, users and other stakeholders. This article provides an overview SE as discussed in the SEBoK and the relationship between SE and systems (for additional information on this, please see Part 2).

Systems and Systems Engineering

In the broad community, the term system “system,” may mean a collection of technical, natural or social elements, or a combination of all three. This may produce ambiguities at times: for example, does “management” refer to management of the SE process, or management of the system being engineered? As with many special disciplines, SE uses terms in ways that may be unfamiliar outside the discipline. For example, in systems science and therefore SE, “open” means that a system is able to interact with its environment--as opposed to being "closed” to its environment. But in the broader engineering world we would read “open” to mean “non-proprietary” or “publicly agreed upon.” In such cases, the SEBoK tries to avoid misinterpretation by elaborating the alternatives e.g. “system management” or “systems engineering management”.

The SEBoK seeks to position SE within the broader scope of knowledge which considers systems as part of its foundations. To do this without attempting to re-define general systems terminology SEBoK introduces two related definitions specific to SE:

- An Engineered System (glossary), is a technical or socio-technical systems system which is the subject of a SE Life Cycle (glossary)
- An engineered System Context (glossary) centres around an engineered system but also includes its relationships other engineered, social or natural systems in one or more defined environments.

Since the province of SE is an engineered systems, most SE literature assumes this in its terminology. Thus, in an SE discussion, “system architecture” would refer to the architecture of the system being engineered (e.g., a spacecraft) and not the architecture of a natural system outside its boundary (e.g., the solar system). In fact, a spacecraft architecture would cover the wider System Context (glossary) including external factors such as changes in gravity and external air pressure and how these affect the spacecraft's technical and human elements. Thus, the term "system architecture" more properly refers to the engineered system context. The SEBoK tries to be more explicit about this, but may still make these kinds of assumption when referring directly to other SE literature.

An extensive glossary of terms identifies how terms are used in the SEBoK, and shows how their meanings may vary in different contexts. As needed, the glossary includes pointers to articles providing more detail.

For more about the definition of systems, see the article What is a System? in Part 2. The primary focus of SEBoK Part 3: Systems Engineering and Management, and Part 4: Applications of Systems Engineering is on how to create or change an engineered system to fulfill the goals of stakeholders within these wider system contexts. The knowledge in Part 5: Enabling Systems Engineering and Part 6: Systems Engineering and Other Disciplines examines the need for SE itself to be integrated and supported within the human activity systems in which it is performed, and the relationships between SE and other engineering and management disciplines.

Scope of Systems Engineering within the Engineered Systems Domain

The scope of SE does not everything involved in the engineer and management of an engineered system. Activities can be part of the SE environment, but other than the specific management of the SE function, not considered to be part of SE. Examples include system construction, manufacturing, funding, and general management. This is reflected in the International Council on Systems Engineering (INCOSE) top-level definition of systems engineering as, “an interdisciplinary approach and means to enable the realization of successful systems.” (INCOSE 2012) Although SE can *enable* the realization of a successful system, if an activity that is outside the scope of SE, such as manufacturing, is poorly managed and executed, SE cannot *ensure* a successful realization.

A convenient way to define the scope of SE within engineering and management is to develop a Venn diagram. Figure 3 shows the relationship between SE, system implementation, and project/systems management. Activities, such as analyzing alternative methods for production, testing, and operations, are part of SE planning and analysis functions. Such activities as production line equipment ordering and installation, and its use in manufacturing, while still important SE environment considerations, stand outside the SE boundary. Note that as defined in Figure 3, system implementation engineering also includes the software production aspects of system implementation. Software engineering, then, is not considered a subset of SE.

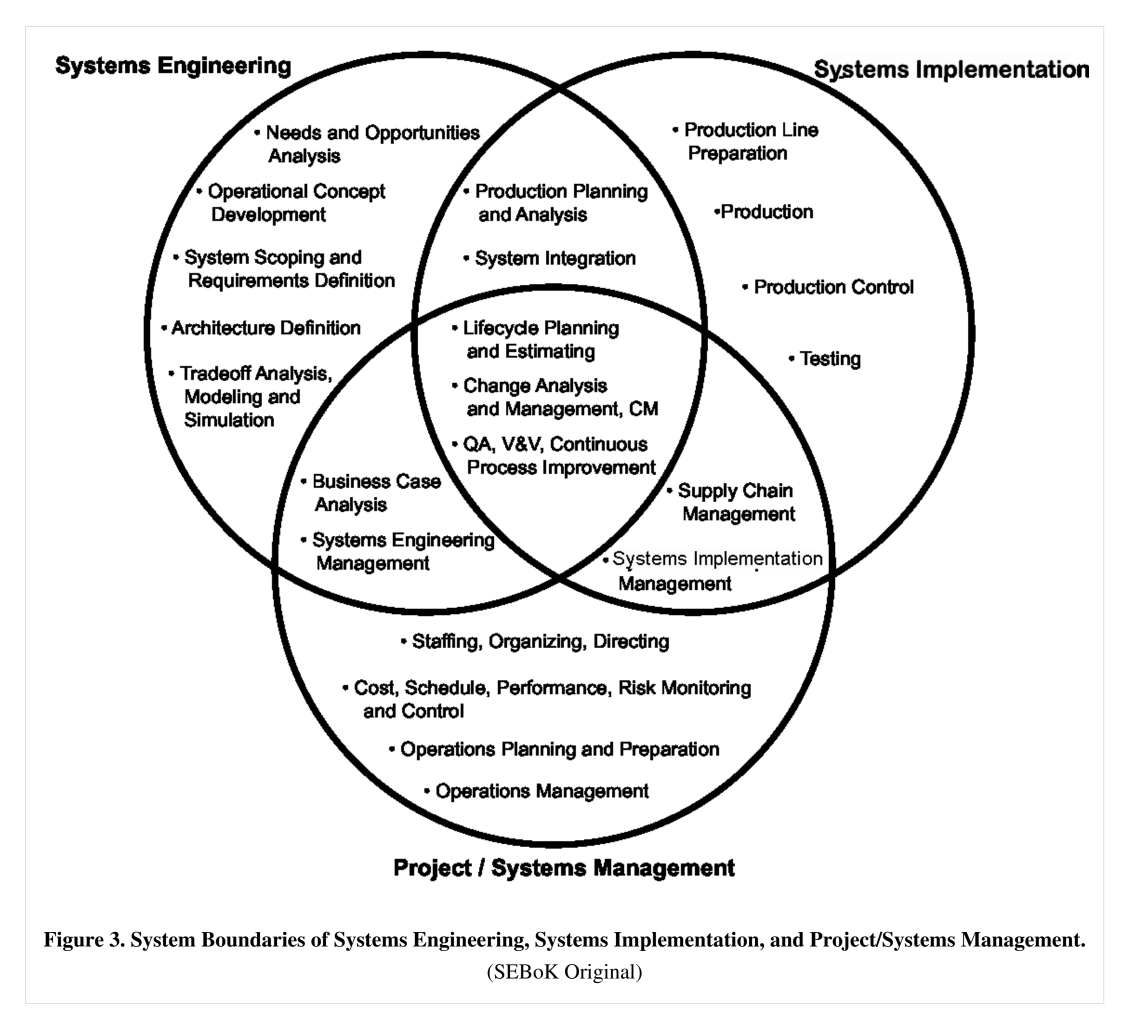


Figure 3. System Boundaries of Systems Engineering, Systems Implementation, and Project/Systems Management. (SEBoK Original)

Traditional definitions of SE have emphasized sequential performance of SE activities, e.g., “documenting requirements, then proceeding with design synthesis …”. (INCOSE 2012) The SEBoK authors depart from tradition to emphasize the inevitable intertwining of system requirements definition and system design in the following revised definition of SE:

> *Systems Engineering (SE) is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on holistically and concurrently understanding stakeholder needs;*

exploring opportunities; documenting requirements; and synthesizing, verifying, validating, and evolving solutions while considering the complete problem, from system concept exploration through system disposal. (INCOSE 2012, modified)

Part 3: Systems Engineering and Management, elaborates on the definition above to flesh out the scope of SE more fully.

Systems Engineering and Engineered Systems Project Life Cycle Context

SE is performed as part of a life cycel approach. Figure 2 summarizes the main agents, activities, and artifacts involved in the life cycle of SE, in the context of a project to create and evolve an engineered system.

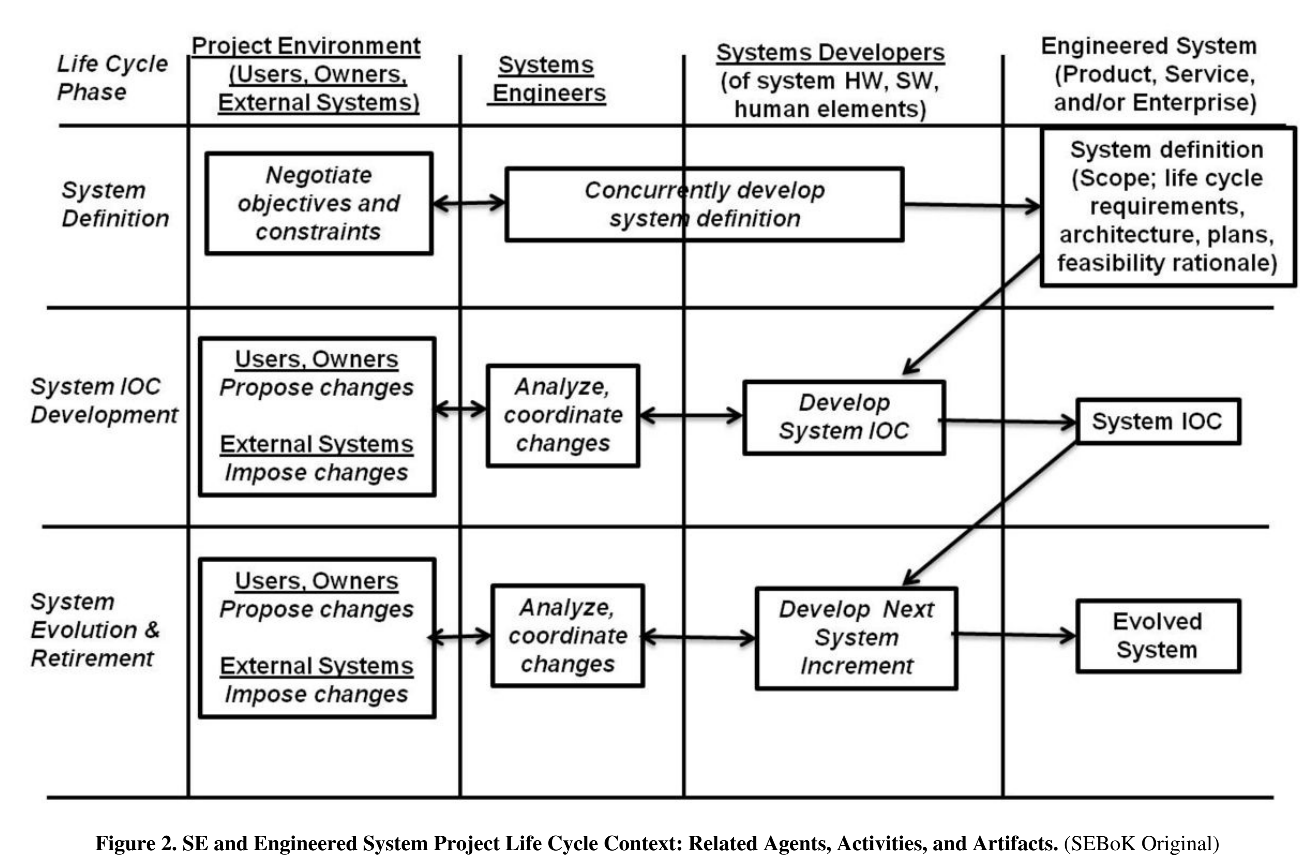


Figure 2. SE and Engineered System Project Life Cycle Context: Related Agents, Activities, and Artifacts. (SEBoK Original)

For each primary project life cycle phase, we see activities being performed by primary agents, changing the state of the ES.

- Primary project life cycle phases appear in the leftmost column. They are system definition, system initial operational capability (IOC) development, and system evolution and retirement.
- Primary agents appear in the three inner columns of the top row. They are systems engineers, systems developers, and primary project-external bodies (users, owners, external systems) which constitute the project environment.
- The ES, which appears in the rightmost column, may be a product, a service, and/or an enterprise.

In each row:

- boxes in each inner column show activities being performed by the agent listed in the top row of that column
- the resulting artifacts appears in the rightmost box.

Arrows indicate dependencies: an arrow from box A to box B means that the successful outcome of box B depends on the successful outcome of box A. Two-headed arrows indicate a two-way dependencies: an arrow that points both from box A to box B and from box B to box A means that the successful outcome of each box depends on the successful outcome of the other.

For example, consider how the inevitable changes that arise during system development and evolution are handled:

- One box shows that the system's users and owners may propose changes.
- The changes must be negotiated with the systems developers, who are shown in a second box.
- The negotiations are mediated by systems engineers, who are shown in a third box in between the first two.
- Since the proposed changes run from left to right and the counter-proposals run from right to left, all three boxes are connected by two-headed arrows. This reflects the two-way dependencies of the negotiation.

An agent-activity-artifact diagram like Figure 1 can be used to capture complex interactions. Taking a more detailed view of the present example demonstrates that:

- The system's users and owners (stakeholders) propose changes to respond to competitive threats or opportunities, or to adapt to changes imposed by independently evolving external systems, such as Commercial-off-the-Shelf COTS products, cloud services, or supply chain enablers.
- Negotiation among these stakeholders and the system developers follows, mediated by the SEs.
- The role of the SEs is to analyze the relative costs and benefits of alternative change proposals, and synthesize mutually satisfactory solutions.

References

Works Cited

Checkland, P. 1981. Systems Thinking, Systems Practice. Hoboken, NJ, USA: Wiley.

INCOSE. 2012. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.

Rechtin, E. 1991. *Systems Architecting*. Upper Saddle River, NJ, USA: Prentice Hall.

Primary References

INCOSE. 2012. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Economic Value of Systems Engineering

The Increasing Value of Systems Engineering

With traditional projects, such as railroads, reservoirs, and refrigerators, a systems engineer faced a self-contained system that typically had relatively stable requirements, a sound scientific base, and numerous previous precedents. As most modern systems become parts within one or more evolving systems of systems (SoS), the performance of effective SE now takes on an ever-higher economic value, as the systems feature a rapidly increasing scale, dynamism, interdependence, human-intensiveness, sources of vulnerability, and novelty.

This is corroborated by the implementation examples in Part 7. Shortfalls in SE lead to either cancellation of already expensive systems or even more expensive systems in terms of total cost of ownership or loss of human life. Part 7 presents the problems in the United States Federal Aviation Administration (FAA) Advanced Automation System (AAS), United States Federal Bureau of Investigation (FBI) Virtual Case File System, the Hubble Space Telescope Case Study, and the Therac-25 medical linear accelerator.

On the other hand, the Global Positioning System (GPS), Miniature Seeker Technology Integration Project (MSTI), and Next Generation Medical Infusion Pump Project all demonstrate that investment in thorough SE results in highly cost-effective systems. Figure 1 summarizes the analyses data by Werner Gruhl, which relates investment levels in SE to cost overruns of the United States National Aeronautics and Space Administration (NASA) projects (Stutzke 2005). The results indicate that there is a general correlation between the amount invested in SE within a program and cost overruns, demonstrating the critical role of properly allocating SE resources.

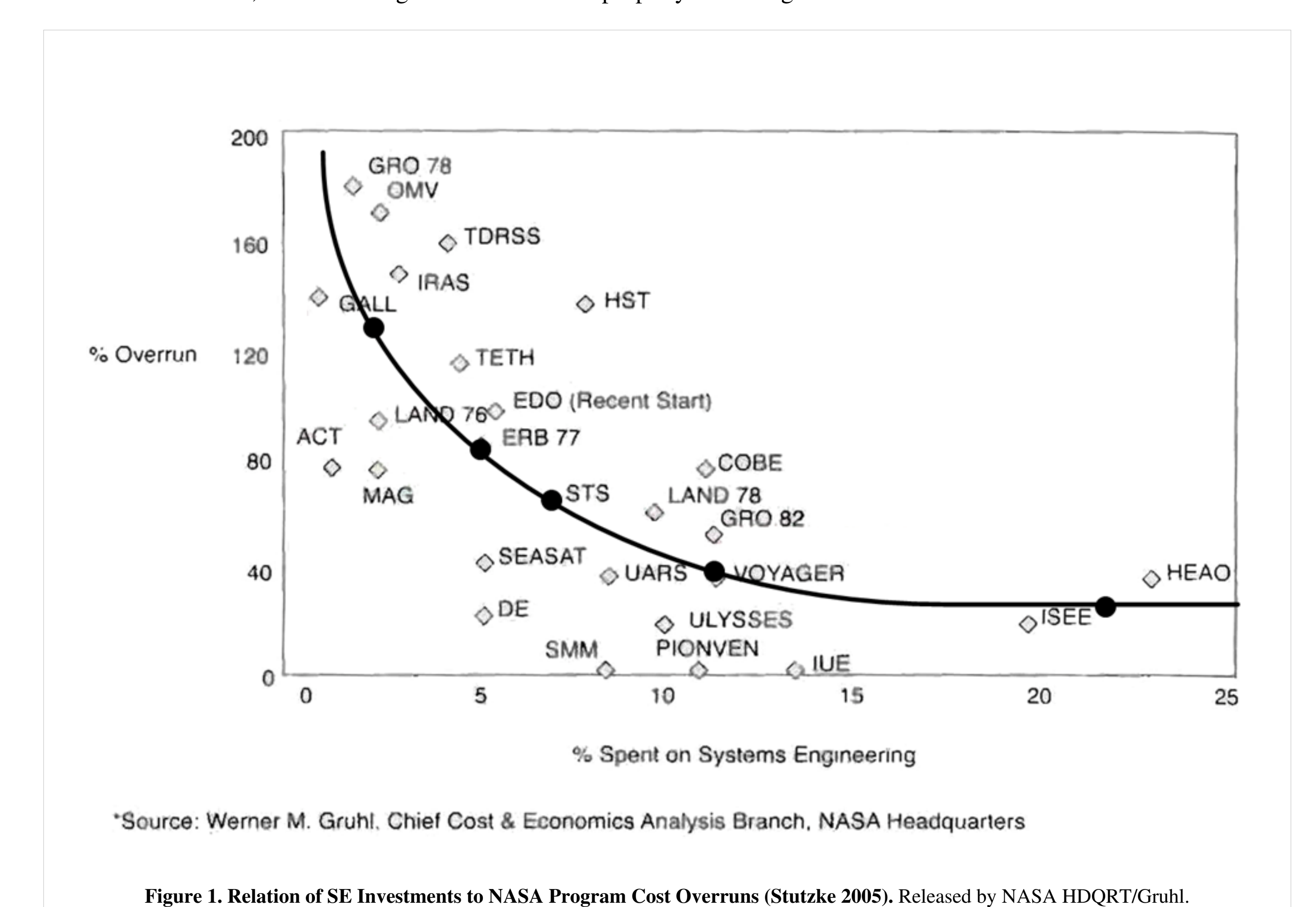


Figure 1. Relation of SE Investments to NASA Program Cost Overruns (Stutzke 2005). Released by NASA HDQRT/Gruhl.

Further Quantitative Evidence of the Value of Systems Engineering

Analysis of the effects of shortfalls in systems architecture and risk resolution (the results of insufficient SE) for software-intensive systems in the 161-project Constructive Cost Model II (COCOMO™ II) database, shows a statistically significant increase in rework costs as a function of project size measured in source lines of code (SLOC): averages of 18% rework for ten-thousand-SLOC projects and 91% rework for ten-million-SLOC projects. This data has influenced many major system projects to reconsider initial underinvestment in SE (e.g., Boehm et al. 2004), and well as to address "how much SE is enough" by balancing the risks of under-investing in SE against those of over-investing (often called "analysis paralysis"), as shown in Figure 2 (Boehm, Valerdi, and Honour 2008).

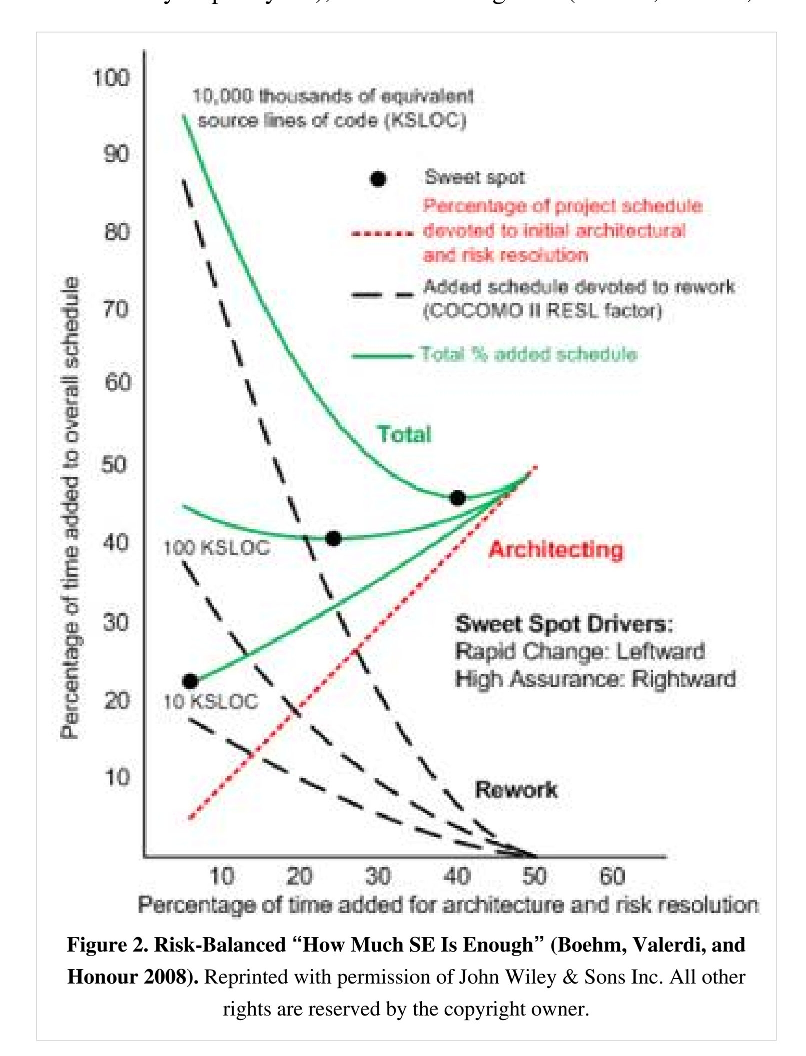


Figure 2. Risk-Balanced "How Much SE Is Enough" (Boehm, Valerdi, and Honour 2008). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Typically, small projects can quickly compensate for neglected SE interface definition and risk resolution; however, as projects grow larger and have more independently-developed components, the cost of late rework negates any savings in reduced SE effort. Additionally, medium-sized projects have relatively flat operating regions, while very large projects pay extremely large penalties for neglecting thorough SE. Extensive surveys and case study analyses corroborate these results.

Survey data on software cost and schedule overruns in *My Life Is Failure: 100 Things You Should Know to Be a Better Project Leader* (Johnson 2006) indicates that the primary sources of the roughly 50% of the commercial projects with serious "software overruns" are the result of shortfalls in SE (lack of user input, incomplete requirements, unrealistic expectations, unclear objectives, and unrealistic schedules). The extensive survey of 46 government-contracted industry projects conducted by the Software Engineering Institute (SEI)/National Defense Industrial Association (NDIA) illustrated a strong correlation between higher project SE capability and higher project performance (Elm et al. 2007). Ongoing research that combined project data and survey data reported in

“Toward an Understanding of The Value of SE” (Honour 2003) and “Effective Characterization Parameters for Measuring SE” (Honour 2010) has provided additional evidence as to the economic value of SE and further insights on critical factors the affect SE success.

A calibrated model for determining “how much SE is enough”, the Constructive Systems Engineering Cost Model (COSYSMO) has been developed and is discussed in (Valerdi 2008). It estimates the number of person-months that a project needs for SE as a function of system size (i.e., requirements, interfaces, algorithms, and operational scenarios), modified by 14 factors (i.e., requirements understanding, technology risk, personnel experience, etc.), which dictates the amount of SE effort needed. Other economic considerations of SE include the costs and benefits of reuse (Wang, Valerdi and Fortune 2010), the management of SE assets across product lines (Fortune and Valerdi 2013), the impact of SE on project risk (Madachy and Valerdi 2010), and the role of requirements volatility on SE effort (Pena and Valerdi 2010).

References

Works Cited

Boehm, B., Brown, A.W., Basili, V., and Turner, R. 2004. "Spiral Acquisition of Software-Intensive Systems of Systems." *CrossTalk*. May, pp. 4-9.

Boehm, B., R. Valerdi, and E.C. Honour. 2008. "The ROI of Systems Engineering: Some Quantitative Results for Software-Intensive Systems." *Systems Engineering*. 11(3): 221-234.

Elm, J. P., D.R. Goldenson, K. El Emam, N. Donatelli, and A. Neisa. 2008. *A Survey of Systems Engineering Effectiveness-Initial Results* (with Detailed Survey Response Data). Pittsburgh, PA, USA: Software Engineering Institute, CMU/SEI-2008-SR-034. December 2008.

Fortune, J., and R. Valerdi. 2013. "A Framework for Systems Engineering Reuse." *Systems Engineering* 16(2).

Honour, E.C. 2003. "Toward An Understanding of The Value of Systems Engineering." Proceedings of the First Annual Conference on Systems Integration, March 2003, Hoboken, NJ, USA.

Honour, E.C. 2010. "Effective Characterization Parameters for Measuring Systems Engineering." Proceedings of the 8th Annual Conference on Systems Engineering Research (CSER). March 17-19, 2010. Hoboken, NJ, USA.

Johnson, J. 2006. *My Life Is Failure: 100 Things You Should Know to Be a Better Project Leader*. Boston, MA, USA: Standish Group International.

Madachy, R., and R. Valerdi. 2010. *Automating Systems Engineering Risk Assessment*. 8th Conference on Systems Engineering Research, Hoboken, NJ.

Pena, M., and R. Valerdi. 2010. "Characterizing the Impact of Requirements Volatility on Systems Engineering Effort." 25th Forum on COCOMO and Systems/Software Cost Modeling, Los Angeles, CA.

Stutzke, R. 2005. *Estimating Software-Intensive Systems*. Boston, MA, USA: Addison Wesley.

Valerdi, R. 2008. *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort in Complex Systems.* Saarbrücken, Germany: VDM Verlag.

Wang, G., R. Valerdi, and J. Fortune. 2010. “Reuse in Systems Engineering,” *IEEE Systems Journal*. 4(3): 376-384.

Primary References

Boehm, B., R. Valerdi, and E.C. Honour. 2008. "The ROI of Systems Engineering: Some Quantitative Results for Software-Intensive Systems." *Systems Engineering*, 11(3): 221-234.

Honour, E.C. 2010. "Effective Characterization Parameters for Measuring Systems Engineering." Proceedings of the 8th Annual Conference on Systems Engineering Research (CSER). March 17-19, 2010. Hoboken, NJ, USA.

Valerdi, R. 2008. *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort in Complex Systems.* Saarbrücken, Germany: VDM Verlag.

Additional References

Hughes, T.P. 2000. *Rescuing Prometheus: Four Monumental Projects that Changed the Modern World*. New York, NY: Vintage Books.

Vanek, F., R. Grzybowski, P. Jackson, and M. Whiting. 2010. "Effectiveness of Systems Engineering Techniques on New Product Development: Results from Interview Research at Corning Incorporated." Proceedings of the 20th Annual INCOSE International Symposium. 12-15 July 2010. Chicago, IL.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Systems Engineering: Historic and Future Challenges

Humans have faced increasingly complex challenges and have had to think systematically and holistically in order to produce successful responses to these challenges. From these responses, generalists have developed generic principles and practices for replicating success. Some of these principles and practices have contributed to the evolution of systems engineering as a discipline.

Historical Perspective

Some of the earliest relevant challenges were in organizing cities. Emerging cities relied on functions such as storing grain and emergency supplies, defending the stores and the city, supporting transportation and trade, afterlife preparations, providing a water supply, and accommodating palaces, citadels, and temples. The considerable holistic planning and organizational skills required to realize these functions were independently developed in the Middle East, Egypt, Asia, and Latin America, as described in Lewis Mumford’s *The City in History* (Mumford 1961).

Megacities, and mobile cities for military operations, such as those present in the Roman Empire, emerged next, bringing another wave of challenges and responses. These also spawned generalists and their ideological works, such as Vitruvius and his *Ten Books on Architecture* (Vitruvius: Morgan transl. 1960). “Architecture” in Rome meant not just buildings, but also aqueducts, central heating, surveying, landscaping, and overall planning of cities.

The Industrial Revolution brought another wave of challenges and responses. In the nineteenth century, new holistic thinking and planning went into creating and sustaining transportation systems, including canal, railroad, and metropolitan transit. General treatises, such as *The Economic Theory of the Location of Railroads* (Wellington 1887), appeared in this period. The early twentieth century saw large-scale industrial enterprise engineering, such as the Ford automotive assembly plants, along with treatises like *The Principles of Scientific Management* (Taylor 1911).

The Second World War presented challenges around the complexities of real-time command and control of extremely large multinational land, sea, and air forces and their associated logistics and intelligence functions. The postwar period brought the Cold War and Russian space achievements. The U.S. and its allies responded to these challenges by investing heavily in researching and developing principles, methods, processes, and tools for military defense systems, complemented by initiatives addressing industrial and other governmental systems. Landmark results included the codification of operations research and SE in *Introduction to Operations Research* (Churchman et. al 1957), Warfield (1956), and Goode-Machol (1957) and the Rand Corporation approach as seen in *Efficiency in Government Through Systems Analysis* (McKean 1958). In theories of system behavior and SE, we see cybernetics (Weiner 1948), system dynamics (Forrester 1961), general systems theory (Bertalanffy 1968), and mathematical systems engineering theory (Wymore 1977).

Two further sources of challenge began to emerge in the 1960s, and accelerated in the 1970s through the 1990s: awareness of the criticality of the human element, and the growth of software functionality in engineered systems.

Concerning awareness of the human element, the response was a reorientation from traditional SE toward “soft” SE approaches. Traditional hardware-oriented SE featured sequential processes, pre-specified requirements, functional-hierarchy architectures, mathematics-based solutions, and single-step system development. A Soft Systems approach to SE is characterized by emergent requirements, concurrent definition of requirements and solutions, combinations of layered service-oriented and functional-hierarchy architectures, heuristics-based solutions, and evolutionary system development. Good examples are societal systems (Warfield 1976), soft systems methodology (Checkland 1981), and systems architecting (Rechtin 1991 and Rechtin-Maier 1997). As with Vitruvius, "architecting" in this sense is not confined to producing blueprints from requirements, but instead extends to concurrent work on operational concepts, requirements, structure, and life cycle planning.

The rise of software as a critical element of systems led to the definition of Software Engineering as a closely related discipline to SE. The Systems Engineering and Software Engineering knowledge area in Part 6: Related Disciplines describes how software engineering applies the principles of SE to the life cycle of computational systems (in which any hardware elements form the platform for software functionality) and of the embedded software elements within physical systems.

Evolution of Systems Engineering Challenges

From 1990 on the rapidly increasing scale, dynamism, and vulnerabilities in the systems being engineered have presented ever-greater challenges. The rapid evolution of communication, computer processing, human interface, mobile power storage and other technologies offers efficient interoperability of net-centric products and services, but brings new sources of system vulnerability and obsolescence as new solutions (clouds, social networks, search engines, geo-location services, recommendation services, and electrical grid and industrial control systems) proliferate and compete with each other.

Similarly, assessing and integrating new technologies with increasing rates of change presents further SE challenges. This is happening in such areas as biotechnology, nanotechnology, and combinations of physical and biological entities, mobile networking, social network technology, cooperative autonomous agent technology, massively parallel data processing, cloud computing, and data mining technology. Ambitious projects to create smart services, smart hospitals, energy grids, and cities are under way. These promise to improve system capabilities and quality of life, but carry risks of reliance on immature technologies or on combinations of technologies with incompatible objectives or assumptions. SE is increasingly needed but increasingly challenged in the quest to make future systems scalable, stable, adaptable, and humane.

It is generally recognized that there is no one-size-fits-all life cycle model that works best for these complex system challenges. Many systems engineering practices have evolved in response to this challenge making use of lean, agile, iterative and evolutionary approaches to provide methods for simultaneously achieving high-effectiveness, high-assurance, resilient, adaptive, and life cycle affordable systems;. The emergence of system of systems (SoS)

approaches have also been introduced, in which independent system elements developed and deployed within their own life cycle are brought together to address mission and enterprise needs.

Creating flexible and tailored life cycles and developing solutions using combinations of engineered systems, each with its own life cycle focus, creates its own challenges of life cycle management and control. In response to this enterprise systems engineering (ESE) approaches have been developed, which consider the enterprise itself as a system to be engineered. Thus, many of the ambitious smart system projects discussed above are being delivered as a program of managed life cycles synchronized against a top down understanding of enterprise needs. It is important that within these approaches we create the flexibility to allow for bottom-up solutions developed by combining open, interoperable system elements to emerge and be integrated into the evolving solutions.

Many of the challenges above, and the SE response to them, increase the breadth and complexity of the systems information being considered. This increases the need for up to date, authoritative and shared models to support life cycle decisions. This has led to the development of model-based systems engineering (MBSE) approaches.

Future Challenges

The INCOSE Systems Engineering Vision 2025 (INCOSE 2014) considers the issues discussed above and from this gives an overview of the likely nature of the systems of the future. This forms the context in which SE will be practised and give a starting point for considering how SE will need to evolve:

- Future systems will need to respond to an ever growing and diverse spectrum of societal needs in order to create value. Individual engineered system life cycles may still need to respond to an identified stakeholder need and customer time and cost constraint. However, they will also form part of a larger synchronized response to strategic enterprise goals and/or societal challenges. System life cycles will need to be aligned with global trends in industry, economy and society, which will, in turn, influence system needs and expectations.
- Future systems will need to harness the ever growing body of technology innovations while protecting against unintended consequences. Engineered system products and services need to become smarter, self-organized, sustainable, resource efficient, robust and safe in order to meet stakeholder demands.
- These future systems will need to be engineered by an evolving, diverse workforce which, with increasingly capable tools, can innovate and respond to competitive pressures.

These future challenges change the role of software and people in engineered systems. The Systems Engineering and Software Engineering knowledge area consider the increasing role of software in engineered systems and its impact on SE. In particular it considers the increasing importance of Cyber-Physical Systems in which technology, software and people play an equally important part in the engineered systems solutions. This requires a SE approach able to understand the impact of different types of technology, and especially the constraints and opportunities of software and human elements, in all aspects of life cycle of an engineered systems.

All of the challenges, and the SE responses to them, make it even more important that SE continues its transition to a model based discipline.

The changes needed to meet these challenges will impact the life cycle processes described in Part 3: Systems Engineering and Management and on the knowledge, skills and attitudes of systems engineers and the ways they are organized to work with other disciplines as discussed in Part 5: Enabling Systems Engineering and Part 6: Related Disciplines. The different ways in which SE is applied to different types of system context, as described in Part 4: Applications of SE, will be a particular focus for further evolution to meet these challenges. The Introduction to SE Transformation knowledge area in SEBoK Part 1 describes how SE is beginning to change to meet these challenges.

References

Works Cited

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. New York, NY, USA: George Braziller.

Checkland, P. 1981. *Systems Thinking, Systems Practice*. Hoboken, NJ, USA: Wiley, 1981.

Churchman, C.W., R. Ackoff, and E. Arnoff. 1957. *Introduction to Operations Research*. New York, NY, USA: Wiley and Sons.

International Council on Systems Engineering (INCOSE), 2014, *Systems Engineering Vision 2025 July*, 2014; Accessed February 16 at http://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf?sfvrsn=4

Ferguson, J. 2001. "Crouching Dragon, Hidden Software: Software in DoD Weapon Systems." IEEE *Software*, July/August, p. 105–107.

Forrester, J. 1961. *Industrial Dynamics*. Winnipeg, Manitoba, Canada: Pegasus Communications.

Goode, H. and R. Machol. 1957. Systems Engineering: An Introduction to the Design of Large-Scale Systems. New York, NY, USA: McGraw-Hill.

McKean, R. 1958. *Efficiency in Government Through Systems Analysis*. New York, NY, USA: John Wiley and Sons.

Mumford, L. 1961. *The City in History*. San Diego, CA, USA: Harcourt Brace Jovanovich.

Rechtin, E. 1991. *Systems Architecting*. Upper Saddle River, NJ, USA: Prentice Hall.

Rechtin, E. and M. Maier. 1997. *The Art of Systems Architecting*. Boca Raton, FL, USA: CRC Press.

Taylor, F. 1911. *The Principles of Scientific Management*. New York, NY, USA and London, UK: Harper & Brothers.

Vitruvius, P. (transl. Morgan, M.) 1960. *The Ten Books on Architecture*. North Chelmsford, MA, USA: Courier Dover Publications.

Warfield, J. 1956. *Systems Engineering*. Washington, DC, USA: US Department of Commerce (DoC).

Wellington, A. 1887. *The Economic Theory of the Location of Railroads*. New York, NY, USA: John Wiley and Sons.

Wiener, N. 1948. *Cybernetics or Control and Communication in the Animal and the Machine*. New York, NY, USA: John Wiley & Sons Inc.

Wymore, A. W. 1977. *A Mathematical Theory of Systems Engineering: The Elements*. Huntington, NY, USA: Robert E. Krieger.

Primary References

Boehm, B. 2006. "Some Future Trends and Implications for Systems and Software Engineering Processes." *Systems Engineering*. Wiley Periodicals, Inc. 9(1), pp 1-19.

INCOSE Technical Operations. 2007. *Systems Engineering Vision 2020*, version 2.03. Seattle, WA: International Council on Systems Engineering, Seattle, WA, INCOSE-TP-2004-004-02.

International Council on Systems Engineering (INCOSE), 2014, *Systems Engineering Vision 2025*, July 2014; Accessed February 16 at http://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf?sfvrsn=4

Warfield, J. 1956. *Systems Engineering*. Washington, DC, USA: US Department of Commerce (DoC). Report PB111801.

Warfield, J. 1976. *Societal Systems: Planning, Policy, and Complexity*. New York, NY, USA: John Wiley & Sons.

Wymore, A. W. 1977. *A Mathematical Theory of Systems Engineering: The Elements*. Huntington, NY, USA: Robert E. Krieger.

Additional References

Hitchins, D. 2007. *Systems Engineering: A 21st Century Methodology.* Chichester, England: Wiley.

The MITRE Corporation. 2011. "The Evolution of Systems Engineering." in *The MITRE Systems Engineering Guide.* Accessed 8 March 2012 at [1].

Sage, A. and W. Rouse (eds). 1999. *Handbook of Systems Engineering and Management*. Hoboken, NJ, USA: John Wiley and Sons, Inc.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1., released 16 October 2018

References

[1] http://www.mitre.org/work/systems_engineering/guide/evolution_systems.html

Systems Engineering and Other Disciplines

As discussed in the Scope of the SEBoK article, there are many touch points and overlaps between systems engineering (SE) and other disciplines. Systems engineers should have a basic understanding of the nature of these other disciplines, and often need to understand aspects of another discipline in detail. This article describes the landscape of disciplines that are intertwined with SE. For a closer view of the individual disciplines, see Part 6.

Engineering Disciplines Other than Systems Engineering

Engineering disciplines are mostly component-oriented and value-neutral in their intellectual content (Boehm and Jain 2006). Their underlying laws and equations, such as Ohm's Law, Hooke's Law, Newton's Laws, Maxwell's equations, the Navier-Stokes equations, Knuth's compendia of sorting and searching algorithms, and Fitts's Law of human movement, pertain to performance in a system-of-interest. They do not address how that performance contributes to the value propositions of stakeholders.

In contrast, SE is more holistic than component-oriented, and more stakeholder value-oriented than value-neutral, performance-oriented in its intellectual content. Realizing successful systems requires reasoning with stakeholders about the relative value of alternative realizations, and about the organization of components and people into a system that satisfies the often-conflicting value propositions of stakeholders. Stakeholders who are critical to the system's success include funders, owners, users, operators, maintainers, manufacturers, and safety and pollution regulators.

In some disciplines, the engineer evaluates and integrates design elements into a system that satisfies proxies of value. The wider the scope of the SoI, the broader the set of SE skills the engineer needs.

For example, an aeronautical engineer might integrate mechanical, electrical, fluid, combustion-chemical, software, and cockpit design elements into a system that satisfies proxies of value like flight range, payload capacity, fuel consumption, maneuverability, and cost of production and maintenance. In so doing, the engineer operates partly as a systems engineer. The SoI is the aircraft itself and the engineer applies aircraft-domain expertise.

However, the same engineer could participate in the engineering of passenger services, airport configurations, baggage handling, and local surface transportation options. All of these contribute to the value propositions of success-critical stakeholders. The SoIs are wider, and the engineer needs broader SE knowledge, skills, and abilities

to operate as a systems engineer. The aircraft-domain expertise remains needed for effective engineering of the wider systems. As discussed in (Guest 1991), most good systems engineers are "T-shaped" people, with both a working knowledge of wider-system considerations, and a deep expertise in a relevant domain, such as aeronautical, manufacturing, software, or human factors engineering.

Engineering disciplines that are intertwined with SE include software engineering (SwE), human factors engineering, and industrial engineering. SwE and SE are not just allied disciplines, they are intimately intertwined (Boehm 1994). Most functionality of commercial and government systems is now implemented in software, and software plays a prominent or dominant role in differentiating competing systems in the marketplace. Software is usually prominent in modern systems architectures and is often the "glue" for integrating complex system components.

The scope of SwE includes both software SE and software construction, but does not include hardware SE. Thus neither SwE nor SE is a subset of the other. See Figure 1 in Scope of the SEBoK. For a definition of the relationship between the SEBoK and the *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, which is published by the Institute of Electrical and Electronics Engineers (IEEE) (Bourque and Fairley 2014.) and is currently under revision, see Systems Engineering and Software Engineering.

Human factors engineering, from micro-ergonomics to macro-ergonomics, is intertwined with SE (Booher 2003; Pew and Mavor 2007). See Human Systems Integration in Part 6.

Industrial engineering overlaps significantly with SE in the industrial domain, but also includes manufacturing and other implementation activities outside of SE. See Systems Engineering and Industrial Engineering in Part 6.

Finally, to field a successful system, a systems engineer may need to know one or more of the many specialty fields in engineering, e.g., security, safety, reliability, availability, and maintainability engineering. Most of these are considered professional disciplines in their own right and many have their own bodies of knowledge. For explanations of how these disciplines relate to SE, overviews of what most systems engineers need to know about them, and references within their bodies of knowledge, see Systems Engineering and Specialty Engineering in Part 6.

Non-Engineering Disciplines

SE is intimately intertwined with two non-technical disciplines: technical management (TM), and procurement and acquisition (also known as acquisition and procurement). TM often falls within the purview of a systems engineer. Many SE textbooks, competency models, and university programs include material about TM. TM is a specialization of project management (PM). SE and PM have significant common content in TM, but neither is a subset of the other. See Figure 1 in the article Scope of the SEBoK. For a definition of the relationship between the SEBoK and the *Guide to the Project Management Body of Knowledge (PMBOK)*, which is published by the Project Management Institute (PMI) (PMI 2013), see Systems Engineering and Project Management in Part 6.

Procurement and acquisition practitioners draw upon SE to determine the scope and overall requirements of the system to be procured or acquired. They then prepare requests for proposals and statements of work, determine evaluation criteria, and design source selection processes. Once a leading source is selected, they decide upon contracting options that encompass payments, reviews, audits, incentive fees, acceptance criteria, procedures, and the nature of deliverables. Finally, they monitor progress with respect to plans (including those for SE), and negotiate and execute changes and corrective actions. Many of these activities amount to specialty disciplines within procurement and acquisition. See the article Related Disciplines in Part 6.

References

Works Cited

Boehm, B. W. "Integrating Software Engineering and Systems Engineering." The Journal of NCOSE Vol. 1 (No. 1): pp. 147-151. 1994

Boehm, B. and A. Jain. 2006. "A Value-Based Theory of Systems Engineering." Proceedings, INCOSE IS 2006. Also available at: http://sunset.usc.edu/csse/TECHRPTS/2006/usccse2006-619/usccse2006-619.pdf.

Booher, H. 2003. *Handbook of Human-Systems Integration*. New York, NY, USA: John Wiley & Sons Inc.

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Guest, D. 1991. "The hunt is on for the Renaissance Man of computing." *The Independent.* London, England: September 17, 1991.

INCOSE. 2011. *Systems Engineering Handbook,* version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.

Pew, R. and A. Mavor. 2007. *Human-System Integration in the System Development Process*. Washington, DC, USA: The National Academies Press.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

Booher, H. 2003. *Handbook of Human-Systems Integration*. New York, NY, USA: John Wiley & Sons Inc.

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Gallagher, B., M. Phillips, K. Richter, and S. Shrum. 2011. *CMMI For Acquisition: Guidelines for Improving the Acquisition of Products and Services,* second ed. Upper Saddle River, NJ, USA: Addison Wesley.

Paulk, M., C. Weber, B. Curtis, and M. Chrissis. 1995. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Upper Saddle River, NJ, USA: Addison Wesley.

Pyster, A. (ed.). 2009. *Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering.* Integrated Software & Systems Engineering Curriculum Project. Hoboken, NJ, USA: Stevens Institute of Technology, September 30, 2009.

Pew, R. and A. Mavor. 2007. *Human-System Integration in the System Development Process*. Washington, DC, USA: The National Academies Press.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Introduction to Systems Engineering Transformation

Introduction to SE Transformation

While the primary focus of the SEBoK is on the current practice of domain independent systems engineering, it is also concerned with the future evolution of the discipline.

The topics in this Knowledge Area (KA) summarize SE knowledge which is emerging and transitioning to become part of the practice of systems engineering, such as Model-Based Systems Engineering (MBSE). In general topics will be introduced here and then expanded into other SEBoK KA's over time.

The knowledge covered in this KA reflects the transformation and continued evolution of SE. For a summary of the current and future challenges that contribute to this evolution, see Systems Engineering: Historic and Future Challenges. This notion of SE transformation and the other areas of knowledge which it includes are discussed briefly below.

Topics

Each part of the SEBoK is divided into Knowledge Areas (KA), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Transitioning Systems Engineering to a Model-based Discipline
- Systems Engineering Core Concepts

Systems Engineering Transformation

The INCOSE Systems Engineering Vision 2025 (INCOSE 2014) describes the global context for SE, the current state of SE practice and the possible future state of SE. It describes number of ways in which SE continues to evolve to meet modern system challenges. These are summarized briefly below.

Systems engineering has evolved from a combination of practices used in a number of related industries (in particular aerospace and defense). These have been used as the basis for a standardized approach to the life cycle of any complex system, see Systems Engineering and Management. Hence, SE practices are still largely based on heuristics. Efforts are under-way to evolve a theoretical foundation for systems engineering, see Foundations of Systems Engineering, considering foundational knowledge from a variety sources.

Systems engineering continues to evolve in response to a long history of increasing system complexity. Much of this evolution is in the models and tools focused on specific aspects of SE, such as understanding stakeholder needs, representing system architectures or modeling specific system properties. The integration across disciplines, phases of development, and projects continues to represent a key systems engineering challenge.

Systems engineering is gaining recognition across industries, academia and governments. However, SE practice varies across industries, organizations, and system types. Cross fertilization of systems engineering practices across industries has begun slowly but surely; however, the global need for systems capabilities has outpaced the progress in systems engineering.

INCOSE Vision 2025 concludes that SE is poised to play a major role in some of the global challenges of the 21st century; that it has already begun to change to meet these challenges and that it needs to undergo a more significant **transformation** to fully meet these challenges. The following bullet points are taken from the summary section of

Vision 2025 and define the attributes of a transformed SE discipline in the future:

- Relevant to a broad range of application domains, well beyond its traditional roots in aerospace and defense, to meet society's growing quest for sustainable system solutions to providing fundamental needs, in the globally competitive environment.
- Applied more widely to assessments of socio-physical systems in support of policy decisions and other forms of remediation.
- Comprehensively integrating multiple market, social and environmental stakeholder demands against "end-to-end" life-cycle considerations and long-term risks.
- A key integrating role to support collaboration that spans diverse organizational and regional boundaries, and a broad range of disciplines.
- Supported by a more encompassing foundation of theory and sophisticated model-based methods and tools allowing a better understanding of increasingly complex systems and decisions in the face of uncertainty.
- Enhanced by an educational infrastructure that stresses systems thinking and systems analysis at all learning phases.
- Practised by a growing cadre of professionals who possess not only technical acumen in their domain of application, but who also have mastery of the next generation of tools and methods necessary for the systems and integration challenges of the times.

Some of these future directions of SE are covered in the SEBoK. Other need to be introduced and fully integrated into the SE knowledge areas as they evolve. This KA will be used to provide an overview of these transforming aspects of SE as they emerge. This transformational knowledge will be integrated into all aspects of the SEBoK as it matures.

References

Works Cited

International Council on Systems Engineering (INCOSE), 2014, *Systems Engineering Vision 2025, July*, 2014; Accessed February 16 at http://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf?sfvrsn=4

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Transitioning Systems Engineering to a Model-based Discipline

Systems engineers have always leveraged many kinds of models, including functional models to support requirements development, simulation models to analyze the behavior of systems, and other analytical models to analyze various aspects of the system such as reliability, safety, mass properties, power consumption, and cost. However, the discipline still relies heavily on document-based artifacts to capture much of the system specification and design information, such as requirements, interface control documentation, and system architecture design descriptions. This information is often spread across many different documents including text, informal drawings, and spreadsheets. This document-based approach to systems engineering suffers from a lack of precision, inconsistencies from one artifact to another, and difficulties in maintaining and reusing the information.

Model-based systems engineering

Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing through development and later life cycle phases (INCOSE 2007). A distinguishing characteristic of a MBSE approach is that the model constitutes a primary artifact of the systems engineering process. The focus on developing, managing and controlling a model of the system is a shift from the traditional document based approach to systems engineering, where the emphasis is on producing and controlling documentation about the system. By leveraging the system model as a primary artifact, MBSE offers the potential to enhance product quality, enhance reuse of the system modeling artifacts, and improve communications among the systems development team. This, in turn, offers the potential to reduce the time and cost to integrate and test the system, and significantly reduce cost, schedule, and risks in fielding a system.

MBSE includes a diverse set of descriptive and analytical models that can be applied throughout the life cycle, and from system of systems (SoS) modeling down to component modeling. Typical models may include descriptive models of the system architecture that are used to specify and design the system, and analytical models to analyze system performance, physical, and other quality characteristics such as reliability, maintainability, safety, and cost.

MBSE has been evolving for many years. The term MBSE was used by Wayne Wymore in his book by this name (Wymore 1993), that provided a state-based formalism for analyzing systems in terms of their input/output characteristics, and value functions for assessing utility of technology independent and technology dependent systems. Simulations have been extensively used across Industry to provide high fidelity performance analysis of complex systems. The Standard for Integration Definition for Function Modeling (IDEF0 1993) was introduced in the 1990’s to support basic functional modeling. A modeling formalism called the enhanced functional flow block diagram (Long 2000) has been used to model many different types of systems. The Object Management Group (OMG) introduced the concept of a Model Driven Architecture (MDA®) (OMG 2003) that leverages a standards-based approach to modeling. The Systems Modeling Language (OMG SysML™) (OMG 2015) was adopted by the OMG in 2006 as a general purpose systems modeling language. In addition, the Unified Profile for DoDAF and MODAF (UPDM) (OMG 2013) was adopted by the OMG in 2008 to support enterprise modeling. Several other domain specific modeling languages have been introduced as well.

MBSE Transition

The INCOSE Systems Engineering Vision 2025 (INCOSE 2014, pg 38) describes the current state of MBSE as follows: 'Model-based systems engineering has grown in popularity as a way to deal with the limitations of document-based approaches, but is still in an early stage of maturity similar to the early days of CAD/CAE.'

SE Vision 2025 also describes a continuing transition of SE to a model-based discipline in which: 'Formal systems modeling is standard practice for specifying, analyzing, designing, and verifying systems, and is fully integrated with other engineering models. System models are adapted to the application domain, and include a broad spectrum of models for representing all aspects of systems. The use of internet driven knowledge representation and immersive technologies enable highly efficient and shared human understanding of systems in a virtual environment that span the full life cycle from concept through development, manufacturing, operations, and support.'

The transition to a more model-based discipline is not without its challenges. This requires both advancements in the practice, and the need to achieve more widespread adoption of MBSE within organizations across industry sectors.

Advancing the practice requires improvements in the modeling languages, methods, and tools. The modeling languages must continue to improve in terms of their expressiveness, precision, and usability. MBSE methods, such as those highlighted in A Survey of Model-Based Systems Engineering (MBSE) Methodologies (Estefan 2008), have continued to evolve, but require further advancements to provide a rigorous approach to modeling a system across the full system lifecycle, while being more adaptable to a diverse range of application domains. The modeling tools must also continue to evolve to support the modeling languages and methods, and to integrate with other multi-disciplinary engineering models and tools in support of the broader model-based engineering effort. The movement towards increased use of modeling standards, that are more widely available in commercial tools, and rigorous model-based methodologies, increase the promise of MBSE.

Many organizations are adopting aspects of MBSE to improve their systems engineering practice, particularly since MBSE was introduced in the INCOSE Systems Engineering Vision 2020 in 2007. However, as indicated in the SE Vision 2025, MBSE is still being applied in pockets within organizations and unevenly across industry sectors. Similar to the evolution of model-based approaches in other disciplines such as mechanical and electrical engineering, the transition occurs incrementally as the methods and tools mature.

The adoption of MBSE requires a workforce that is skilled in the application of MBSE. This requires organizations to provide an infrastructure that includes MBSE methods, tools, and training, and a commitment to deploy this capability to their programs As with any organizational change, this must be approached strategically to grow this capability and learn from their experiences.

Like other engineering disciplines, the transition of systems engineering to a model-based discipline is broadly recognized as essential to meet the challenges associated with increasing system complexity, and achieving the productivity and quality improvements. The SEBoK will continue to reflect the growing body of knowledge to facilitate this transition.

References

Works Cited

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies*, rev, B. Seattle, WA: International Council on Systems Engineering.

INCOSE-TD-2007-003-02. Accessed April 13, 2015 athttp:/ / www. omgsysml. org/ MBSE_Methodology_Survey_RevB.pdf

INCOSE Technical Operations. 2007. *Systems Engineering Vision 2020*, version 2.03. Seattle, WA: International Council on Systems Engineering, Seattle, WA, INCOSE-TP-2004-004-02.

International Council on Systems Engineering (INCOSE), 2014,S*ystems Engineering Vision 2025July*, 2014; Accessed February 16 at http://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf?sfvrsn=4

Long , James E., 2000,*Systems Engineering (SE) 101, CORE ®: Product & Process Engineering Solutions , Vitech training materials,*Vitech Corporation , Vienna, VA.

Object Management Group (OMG), 2003, *Model-Driven Architecture (MDA) Guide, v1.01, June 12 2003*, available at http://www.omg.org/mda/.

Object Management Group (OMG), 2015, *OMG Systems Modeling Language (OMG SysML™), V1.4*, OMG document number formal/2015-06-03, September 2015; http://www.omg.org/spec/SysML/1.4/

Object Management Group (OMG), 2013, *Unified Profile for DoDAF/MODAF (UPDM) OMG document* number formal/2013-08-04, August 2013, http://www.omg.org/spec/UPDM/2.1/

Standard for Integration Definition for Function Modeling(IDEF0), 1993, Draft Federal Information Processing Standards, Publication 183, December 21, 1993.

Wymore, W.,*Model-Based Systems Engineering*. Boca Raton, FL: CRC Press , 1993

Primary References

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities* [1], Section 9.2 Model-Based Systems Engineering, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Additional References

MBSE Wikipedia. Accessed February 16, 2015, http://www.omgwiki.org/MBSE/doku.php

---

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://sebokwiki.org/draft/INCOSE_Systems_Engineering_Handbook

Systems Engineering Core Concepts

Purpose and Uses of the Systems Engineering Concept Model

The Systems Engineering Concept Model (SECM) captures the concepts that are referred to in the Systems Engineering Body of Knowledge. The SECM provides a means to evaluate the consistency and coverage across the broad set of Systems Engineering concepts, and can facilitate communication to better understand and evolve these concepts. Although the primary reference for the SECM was the SEBoK, the concepts were cross-checked against other industry references including ISO/IEC/IEEE 15288:2015 and the INCOSE Systems Engineering Handbook Version 4, enabling the SECM to be used to evaluate, understand, and evolve the concepts in these references as well. A small team developed the SECM to support the requirements for the next generation of the OMG Systems Modeling Language (OMG SysML™) to ensure SysML is consistent with the three leading industry standards.

SECM Approach

A concept model, called the Concept Map, was developed by the original SEBoK team prior to release of the SEBoK v1.0 in 2012, and was used to support integration of the initial concepts across the SEBoK topic areas. The Concept Map included a concept model, and a mapping of the concepts to the glossary terms and to the sections of the SEBoK.

The SECM captures the Systems Engineering concepts and their relationship that are contained in today's SEBoK. The small subset of UML constructs and symbols, shown in Figure 1, are used to represent the SECM model. The choice of notations is intended to balance simplicity, understandability, and precision.

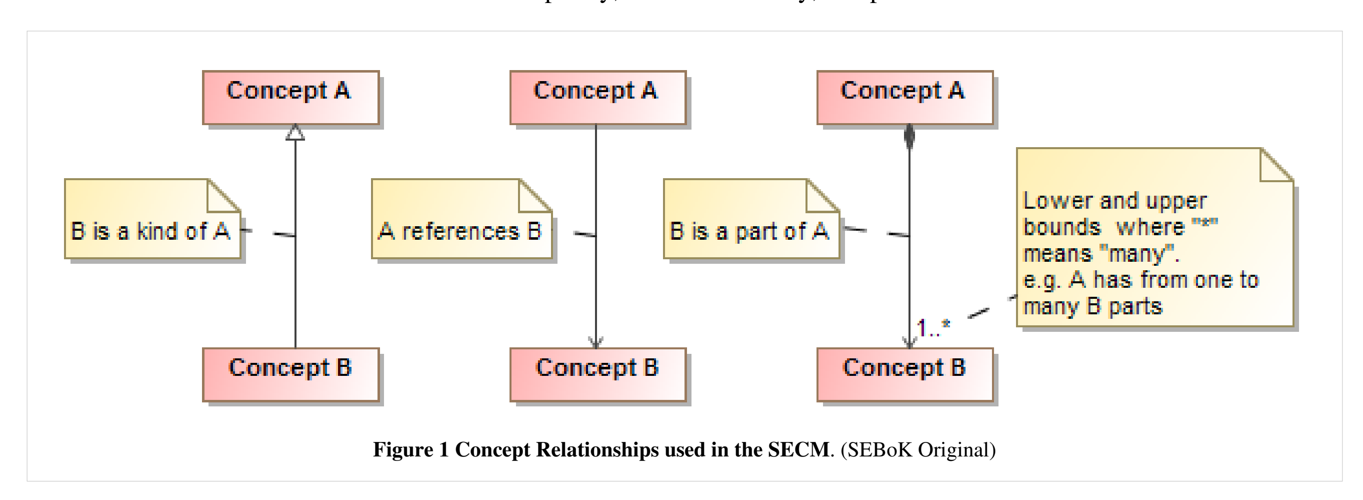


Figure 1 Concept Relationships used in the SECM. (SEBoK Original)

Figure 2 below shows a usage of these constructs and symbols when applied to a simple example of a Car. This diagram shows that a Car is kind of a Vehicle, and that the Driver drives the car (a reference relationship), and that there are four wheels that are parts of the Car.

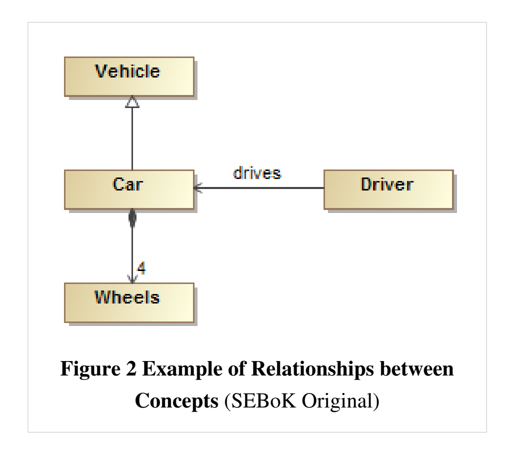


Figure 2 Example of Relationships between Concepts (SEBoK Original)

The SECM is presented in a series of diagrams that generally represent concepts for particular knowledge areas and topics in the SEBoK, and also include references to glossary terms in the SEBoK.

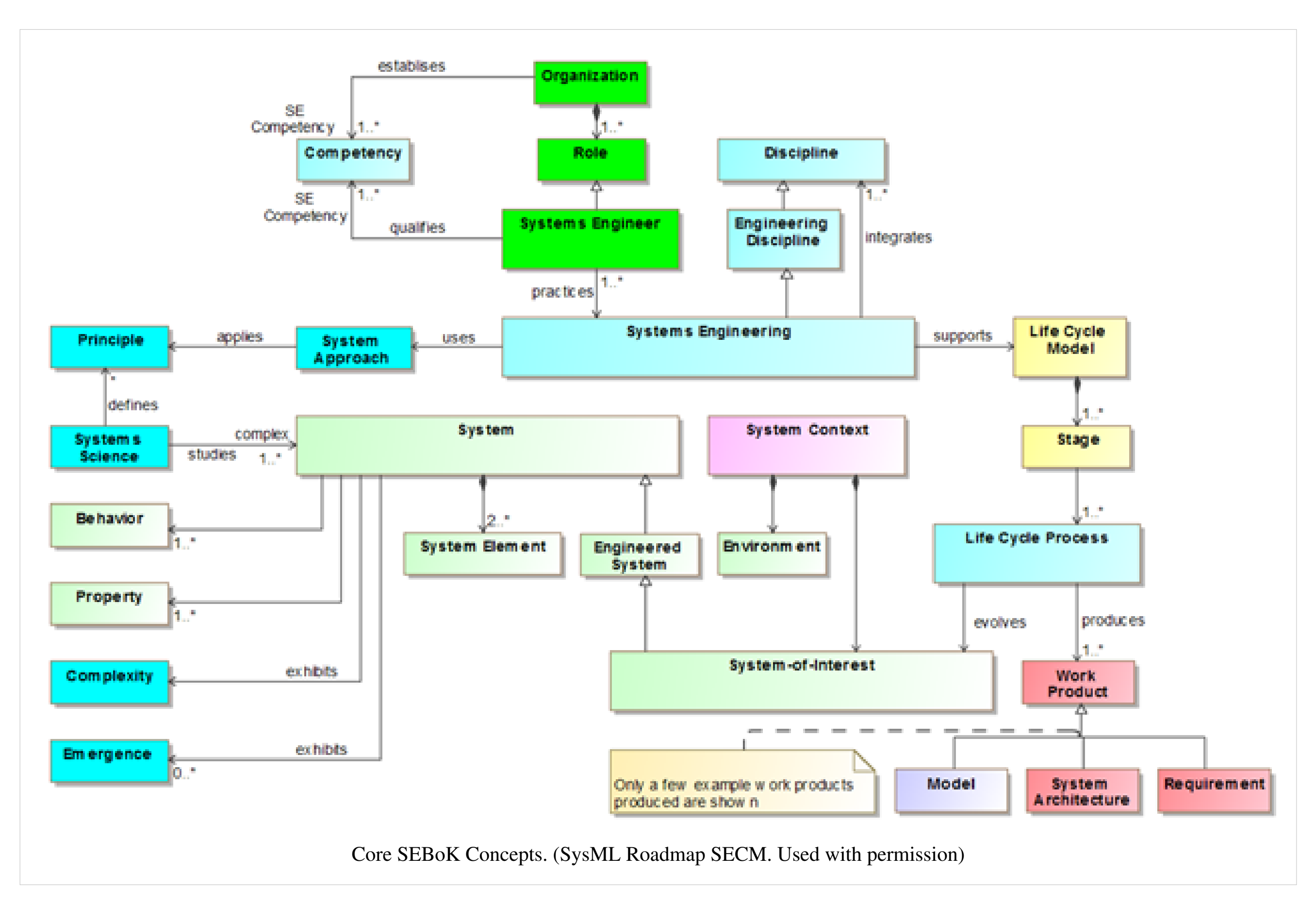
The approach used to capture the SECM content from the SEBoK is described in the following steps:

1. A topic within a SEBoK knowledge area is selected and the text is evaluated to identify key Systems Engineering concepts
2. From the sentences containing the concepts, the subjects, i.e. the concepts, and the predicate, i.e. the relationship between concepts, are identified. The predicate is a statement that says something about the subject
3. A search is conducted for this concept in other areas of the SEBoK, and the accompanying text, is evaluated to further refine the concept and its relationships to other concepts.
4. The definition of the concept in the SEBoK glossary, if available, is evaluated.
5. The use and definition of this concept in the other two industry references is evaluated.
6. The concept and a derived definition from the above evaluation are added to the SECM.
7. The discovered relationships between the concepts are added to the model and to the relevant diagrams. The diagrams group related concepts often associated with a SEBoK Knowledge Area or Topic.
8. As new contributions are made to the SEBoK, this approach can be used to identify new concepts and relationships, and add them to the SECM.

Introduction to Core Concepts

The SECM was developed independently of the BKCASE project to support the evolution of the SysML standard. These models can be used to identify inconsistencies and/or areas requiring additional coverage within the SEBoK to support future updates.

The Core Concept Diagram shown in Figure 3 provides a high level view of the some of the key concepts presented in the SEBoK. It should be emphasized that this figure is intended to be a representative interpretation of the current SEBoK without modifying or adding concepts, and no claim of the completeness of the SEBoK concepts is made. The colors correspond to logical groupings of concepts.



Core SEBoK Concepts. (SysML Roadmap SECM. Used with permission)

A brief description of the core concepts model is provided below.

A *Systems Engineer* is a role within an *Organization* that practices the *Engineering Discipline of Systems Engineering* (SE), and is qualified by a set of SE *Competencies*. *Systems Engineering* integrates other *Disciplines* to support the *Life Cycle Model*. The Life Cycle Model is composed of life cycle *Stages* that typically include conceptual, realization, production, support, utilization and retirement stages (not shown). Each life cycle *Stage* refers to *Life Cycle Processes* that produce various kinds of *Work Products*. The *Life Cycle Processes* evolve the *System-of-Interest* (SoI).

There are many kinds of systems including natural systems, social systems, and technological systems (not shown). Systems that are created by and for people are referred to as *Engineered Systems*. An *Engineered System* whose life cycle is under consideration is referred to as a System-of-Interest (SoI).

A System-of-Interest is part of a broader *System Context*, which also includes an *Environment*. The Environment consists of other open systems (not shown) that can influence the SoI. Systems are composed of System Elements, and have Behavior and Properties. Systems can exhibit *Emergence* and *Complexity*.

Systems Engineering uses a *System Approach* which applies established system *Principles*. *Systems Science* is an interdisciplinary field of science that studies complex systems and helps define and update the *Principles* that are applied by the *System Approach*, which is used by the discipline of Systems Engineering.

References

Works Cited

ISO/IEC/IEEE. 2015. Systems and Software Engineering -- System Life Cycle Processes. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

INCOSE. 2015. Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

SysML Roadmap: Systems Engineering Concept Model Workgroup (SECM) [OMG SysML Portal] http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap%3Asystems_engineering_concept_model_workgroup

Primary References

ISO/IEC/IEEE. 2015. Systems and Software Engineering -- System Life Cycle Processes. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

INCOSE. 2015. Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: SEBoK Users and Uses

SEBoK Users and Uses

The users and uses described in this article were identified based on the six SEBoK purposes described in the SEBoK Introduction.

Users can either be primary (those who use the SEBoK directly) or secondary (those who use the SEBoK with assistance from a systems engineer). Indicative, but not exhaustive, sets of example uses are shown in Tables 1 and 2 below.

New to SEBoK or Systems Engineering?

The list of users and use cases below allow someone who has come to the SEBoK with a particular focus to identify quickly where to focus their reading. If you are completely new to systems engineering or have no clear view of how it is covered in the SEBoK you should use Use Case 0 below to orient yourself and learn the basics before looking at the other use cases:

- Use Case 0: Systems Engineering Novices

Primary Users

Primary users are those who use the SEBoK directly, as shown in Table 1. Hyperlinks in the second column link to the associated use case, where one has been written. The use cases are listed at the end of the topic, and may also be seen here. [1]

Table 1. Primary SEBoK Users and Common Uses. (SEBoK Original)

#	Users	Uses
1	Practicing Systems Engineers ranging from novice through expert	• Taking on a new SE role in a project; preparing by finding references for study • Expanding SE expertise and specialization; preparing by finding references for study • Seeking to understand the principles of SE; seeking the best references to elaborate on those principles • Reviewing a project or mentoring a new SE performer; seeking to understand what best practices to look for • Pursuing professional development through study of SE topics, including new developments in SE
2	Process engineers responsible for defining or implementing SE processes	• Maintaining a library of SE process assets; seeking to understand which SE process models and standards are most relevant • Tailoring a process for a specific project; seeking to learn how others have tailored processes, or how a specific application domain affects tailoring • Measuring the effectiveness of an organization's SE processes; seeking to learn how others have done that • Defining standards for a professional society or standards organization
3	Faculty Members	• Developing a new graduate program in SE, and deciding what core knowledge all its students must master; the user should consult the *Graduate Reference Curriculum for Systems Engineering (GRCSE™)* in conjunction with the SEBoK • Developing a new SE course; seeking to identify course objectives, topics, and reading assignments • Incorporate SE concepts in courses or curricula focused on engineering disciplines other than SE
4	GRCSE authors	• As members of the GRCSE author team, deciding what knowledge to expect from all SE graduate students • See *Graduate Reference Curriculum for Systems Engineering (GRCSE™)* (Pyster et al. 2015)

5	Certifiers	• Defining a company's in-house SE certification program; seeking to understand what others have done, how such programs are typically structured, and how to select the knowledge that each person seeking certification should master • Defining certification criteria for a professional society or licensure program
6	General Managers, Other Engineers, developers, testers, researchers	• Mastering basic vocabulary, boundaries, and structure of SE; seeking a few primary references • Learning what the scope of SE is, relative to the General Manager role • Learning what the role of the systems engineer consists of, relative to others on a project or in an organization • Learning to effectively perform a general manager role on an SE integrated product team
7	Customers of Systems Engineering	• Providing resources to and receiving artifacts from systems engineers • Seeking to better understand what to ask for, how to request it, how much to pay for it, and how to judge the quality of what is received
8	SE managers	• Evaluating possible changes in team processes and tools proposed by systems engineers on various teams; seeking independent information with which to evaluate the proposals • Hiring systems engineers, and developing competency-based job descriptions
9	SE researchers	• Looking for gaps are in SE knowledge to help guide a research agenda • Getting familiarize with a research topic; seeking the most important articles about the topic

Secondary Users

Secondary users are those who use the SEBoK with assistance from a systems engineer, as shown in Table 2.

Table 2. Secondary SEBoK Users and Common Usages. (SEBoK Original)

#	Users	Uses
1	Human resource development professionals	• Supporting the hiring and professional development of systems engineers
2	Non-technical managers	• Augmenting understanding of central concerns with information about relevant SE topics; e.g., a contracting manager might want to better understand SE deliverables being called out in a contract
3	Attorneys, policy makers	• Defining the impact of SE performance on central concerns; e.g., understanding the liability of a systems engineer for errors in judgment on a project, or the limitations of SE in guaranteeing the success of a project against actions of sponsors, managers, or developers

List of Use Cases

At this time not every class of user has a use case developed. To illustrate the major uses, the following use cases are included:

- Use Case 1: Practicing Systems Engineers. This covers the first set of users from Table 1.
- Use Case 2: Other Engineers. This covers the second and sixth sets of users from Table 1.
- Use Case 3: Customers of Systems Engineering. This covers the seventh set of users from Table 1.
- Use Case 4: Educators and Researchers. This covers the third, fourth, and ninth sets of users from Table 1.
- Use Case 5: General Managers. This covers the sixth and eighth sets of users from Table 1.

While not exhaustive, the use cases show the utility of the SEBoK in various applications and contexts.

References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://sebokwiki.org/draft/Case_Studies

Use Case 0: Systems Engineering Novices

Some users of the Systems Engineering Body of Knowledge (SEBoK) may be new to the field. This article provides recommended readings for such a user.

Learn the Basic Terms

As discussed in the Introduction to the SEBoK, there are four key terms that you should first understand when learning about systems engineering (SE):

- A system is “a collection of elements and a collection of inter-relationships amongst the elements such that they can be viewed as a bounded whole relative to the elements around them. Open Systems exists in an environment described by related systems with which they may interact and conditions to which they may respond. While there are many definitions of the word “system,” the SEBoK authors believe that this definition encompasses most of those which are relevant to SE.
- An engineered system is an open system of technical or sociotechnical elements that exhibits emergent properties not exhibited by its individual elements. It is created by and for people; has a purpose, with multiple views; satisfies key stakeholders’ value propositions; has a life cycle and evolution dynamics; has a boundary and an external environment; and is part of a system-of-interest hierarchy.
- Systems engineering is “an interdisciplinary approach and means to enable the realization of successful (engineered) systems”. It focuses on holistically and concurrently understanding stakeholder needs; exploring opportunities; documenting requirements; and synthesizing, verifying, validating, and evolving solutions while considering the complete problem, from system concept exploration through system disposal.
- A systems engineer is “a person who practices systems engineering” as defined above, and whose systems engineering capabilities and experience include sustained practice, specialization, leadership, or authority over SE activities. These activities may be conducted by any competent person regardless of job title or professional affiliation.

Get an Overview

The next step for someone new to SE is get an overview of the discipline. Part 1: SEBoK Introduction contains four articles particularly helpful to one new to SE.

- The article Systems Engineering Overview frames systems engineering inside the larger topic of 'Systems Science.'
- The article Economic Value of Systems Engineering makes the business case for investing in systems engineering as a way to reduce total ownership cost.
- The article Systems Engineering and Other Disciplines discusses briefly how systems engineers and other engineers interact as together they develop complex systems.
- Finally, the article Systems Engineering: Historic and Future Challenges gives a quick history of the discipline and discusses what lays ahead.

Learn about Systems

Engineering is often described as the application of science to develop new products or systems. Part 2: Foundations of Systems Engineering describes some of the underlying systems principles that form the foundation for systems engineering.

- The Knowledge Area on Systems Fundamentals contains five articles. What is a System? is recommended for a new user.
- The Knowledge Area on Systems Science presents two articles on its history and approaches. Both are recommended.
- The Knowledge Area on Systems Thinking has four articles. The first, What is Systems Thinking?, is recommended on a first reading.
- One of the most important current research and practice areas of SE is Model Based Systems Engineering (MBSE). The Knowledge Area Representing Systems with Models provides the foundation for MBSE. The first three of the five articles in the KA are recommended.

Learn how the Systems Approach is Applied to Engineered Systems

The Knowledge Area Systems Approach Applied to Engineered Systems describes how systems science and systems thinking lead to the practice of systems engineering. All eight articles are recommended.

- Overview of the Systems Approach
- Engineered System Context
- Identifying and Understanding Problems and Opportunities
- Synthesizing Possible Solutions
- Analysis and Selection between Alternative Solutions
- Implementing and Proving a Solution
- Deploying, Using, and Sustaining Systems to Solve Problems
- Stakeholder Responsibility
- Applying the Systems Approach

Explore the Methods of Systems Engineering

The SEBoK uses a life-cycle framework to describe the processes that comprise systems engineering. Part 3: SE and Management contains the plurality of the content of the SEBoK in eight knowledge areas. A new user should be familiar with the introductions to each of these Knowledge Areas, and should read further in those KAs of interest.

- Life Cycle Models
- Concept Definition
- System Definition
- System Realization
- System Deployment and Use
- Systems Engineering Management
- Product and Service Life Management
- Systems Engineering Standards

Explore the Applications of Systems Engineering

The SEBoK partitions the body of knowledge between methods and areas of application. Areas of application are classified as:

- Product Systems Engineering
- Service Systems Engineering
- Enterprise Systems Engineering
- Systems of Systems (SoS)

A new user should read the introduction to Part 4: Applications of Systems Engineering and to the four knowledge areas listed above. The reader's interests can then suggest which further reading should be done.

Read Case Studies

Finally, the new user should scan the case studies and vignettes in Part7: SE Implementation Examples and read a few of those in areas that appeal to the reader. This will help reinforce the fundamentals as well as illustrate the practice of SE.

The following case studies are included:

- Successful Business Transformation within a Russian Information Technology Company
- Federal Aviation Administration Next Generation Air Transportation System
- How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn
- Hubble Space Telescope Case Study
- Global Positioning System Case Study
- Medical Radiation Case Study
- FBI Virtual Case File System Case Study
- MSTI Case Study
- Next Generation Medical Infusion Pump Case Study

For Later Reading

Part 6: Related Disciplines contains a broad selection of Knowledge Areas and Topics that describe how systems engineers work with other disciplines. The Knowledge area on SE and Software Engineering is particularly important, as modern systems get much of their functionality from software.

Part 5: Enabling Systems Engineering has KAs describing how individuals, teams, and organizations can develop to practice effective systems engineering.

A person new to SE should become familiar with several references that are beyond that SEBoK. They include the INCOSE Handbook, several standards (listed in Relevant Standards), and the main journals of systems engineering (including but not limited to *Systems Engineering*, the *Journal of Enterprise Transformation*, and *Systems, Man, and Cybernetics*).

References

Works Cited

None.

Primary References

None.

Additional References

None.

Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Use Case 1: Practicing Systems Engineers

Both for the entry-level systems engineer learning the discipline of systems engineering (SE), and the more experienced systems engineer seeking the knowledge required to accomplish a work activity, the SEBoK serves as a primary information source and a quick, comprehensive reference for SE information.

What these system engineers find in the SEBoK includes:

- definitions of terms,
- explanations of basic concepts and principles,
- useful discussions of topics,
- references to articles and textbooks that cover topics in-depth, and
- pointers to additional sources.

How Systems Engineers Use Topics

Researching SE-related subjects, identifying educational resources, and connecting with individuals or organizations which offer specialized expertise are all part of the job for the practicing systems engineer. The time available to the SE for these activities can be quite limited. The SEBoK is designed to ease the pressure on the systems engineer in this situation, in several ways:

- Because its content is based on research, proven practices, and emerging knowledge, the SEBoK makes high-quality information available to the systems engineer right away.
- Being composed of articles of 2000 words or less in most cases, the SEBoK enables the systems engineer to quickly get an overview of relevant topics.
- By providing primary references, each topic offers a direct route to more detailed information.
- Even greater detail, breadth, and a sense of what's relevant in the SE literature are available through the additional references each topic provides.
- Since the SEBoK sources have been reviewed and vetted by a team of experts, the SEBoK helps the systems engineer avoid less reliable information which can be hard to eliminate within Internet search results.
- The systems engineer who needs to connect with educators and researchers can find relevant names and institutions in SEBoK topics and references.

Systems engineers using the SEBoK may choose one or more of several approaches:

- searching on keywords or article names, using the text field, Search [1] button, and Go [2] button at the top right of each SEBoK page
- scanning the Quick Links, Outline (where the table of contents is located), or Navigation indexes that appear at the left of each SEBoK page, and following links from there to articles that seem likely to be of interest
- searching on keywords using an Internet search engine
- reading through one or more of Parts 1 through 7 in sequence

Reading the SEBoK in sequence is especially suitable for the practicing engineer who is new to SE, or is enrolled in an SE-related training course. For this engineer, SE (or some aspect of it) is a subject to be learned comprehensively. This is made easier by navigation links from each article to the previous, next, and parent articles as found in the Table of Contents.

For practicing systems engineers, having the SEBoK makes it possible to gain knowledge more quickly and reliably than they would otherwise. The goal is to spend less time searching for and compiling new information from disparate sources and more time getting work done.

For a team of practicing engineers, the gap in knowledge between more- and less-experienced engineers can be a major obstacle. The SEBoK serves as a tool for the team to build a framework of agreed-upon definitions and perspectives. The consistency of such a framework enhances communication across the team. New teams, especially,

can benefit from bridging the gap between legacy and more-recently-acquired knowledge. For more information, see Enabling Teams in Part 5.

How Systems Engineers Use Implementation Examples

The SEBoK is written, for the most part, independent of any particular domain of practice. By design, parts 1 though 6 focus on the discipline of SE and not the numerous domains where SE can be applied.

This lack of domain-specific content is partly offset by Part 7, Systems Engineering Implementation Examples, which consists of case studies and examples drawn from a number of domains where SE is applied. Each example demonstrates the impact of a particular application domain upon SE activities. Examples are generally most useful to the systems engineer when they are aligned with the domain in which the he or she is working, but sometimes ideas from an example in one domain can be usefully applied to situations in another.

Example: Model-Based Systems Engineering Practitioners

For practitioners of model-based systems engineering (MBSE), the Representing Systems with Models knowledge area is of central importance within the SEBoK.

Academic faculty who use the SEBoK to support curriculum development and assessment can refer to the same knowledge area to ensure that their curricula accurately cover the languages and/or methodologies such as System Modeling Language (SysML) and Object-Process Methodology (OPM).

SE researchers, too, can adopt an MBSE approach, making their research products more formal and rigorous by basing them on models.

In MBSE, models of systems support system life cycle activities, including requirements engineering, high-level architecture, detailed design, testing, usage, maintenance, and disposal.

Vignette: Systems Engineering for Medical Devices

Tara Washington has worked as a engineer for the HealthTech medical device company for seven years. Besides continuing to improve her strong software skills, she has shown an aptitude for systems thinking. To better understand the products that her software supports, Tara has taken courses in electrical engineering, mechanical engineering, and physiology. The coursework has helped her to perform effectively as a software system analyst on the SE teams of her last two projects.

HealthTech's Research Division proposes a new concept for a highly programmable radiation therapy device that monitors the effects of the radiation on various parts of the body and adjusts the parameters of the radiation dosage to maximize its effectiveness, subject to a number of safety constraints. The software-intensiveness of the device leads Tara's project manager to recommend her as the lead systems engineer for the design and development of the product.

Tara welcomes the opportunity, knowing that she possesses enough domain knowledge to take the lead SE role. Even so, she realizes that she has picked up SE skills mainly by intuition and needs to build them up more systematically. Tara begins to consult some of HealthTech's lead systems engineers, and to study the SEBoK.

After reading the SEBoK Introduction, Tara feels that she has a solid overview of the SEBoK. Tara finds that the next topic, Scope and Context of the SEBoK, outlines the key activities that she expects to lead, along with others which will require her to collaborate with systems developers and project and systems management personnel.

The same topic identifies those parts of the SEBoK that Tara needs to study in preparation for her lead systems engineer role:

- SE concepts, principles, and modeling approaches in Part 2 (Representing Systems with Models knowledge area (KA))

- life cycle processes, management, technical practices, in Part 3 (Systems Engineering and Management KA)
- approaches for specifying, architecting, verifying and validating the hardware, software, and human factors aspects of the product, as well as common pitfalls to avoid and risks to manage, also in Systems Engineering and Management
- guidelines for the systems engineering of products, in Part 4: Applications of Systems Engineering, including references
- SE knowledge, skills, abilities, and attitudes (KSAAs) needed for a project in Part 5: Enabling Systems Engineering including references
- specialty engineering disciplines that may be key to the project's success, in Part 6: Related Disciplines

Tara's awareness of the deaths caused by the Therac-25 radiation therapy device motivates her to study not only the Safety Engineering topic in Part 6, but all of its key references as well.

While reading about SE life cycle process models in Systems Engineering and Management in Part 3, Tara notes the reference to the Next Generation Medical Infusion Pump Case Study in Part 7. This case study strikes Tara as highly relevant to her medical-device work, and she observes that it is organized into phases similar to those used at HealthTech. From the case study, Tara gains understanding of how a project such as hers would progress: by concurrently evaluating technology opportunities, by discovering the needs of various device stakeholders such as patients, nurses, doctors, hospital administrators, and regulatory agencies, and by working through increasingly detailed prototypes, specifications, designs, plans, business cases, and product safety analyses.

The case study mentions its source: Human-System Integration in the System Development Process [3] (Pew and Mavor 2007), published by the U.S. National Research Council. Tara obtains this book. In it, she finds numerous good practices for human-systems needs analysis, organizational analysis, operations analysis, prototyping, usability criteria formulation, hardware-software-human factors integration, process decision milestone review criteria, and risk management.

As a result of her SEBoK-based study, Tara feels better-qualified to plan, staff, organize, control, and direct the SE portion of the HealthTech radiation therapy device project and to help bring the project to a successful conclusion.

Summary

In the SEBoK, practicing engineers have an authoritative knowledge resource that can be accessed quickly to gain essential high-level information, and to identify the best references for in-depth study and research into SE topics when an individual's initial level of understanding is not adequate to get the job done.

The SEBoK is also a resource for practicing engineers who teach, as well as those taking training courses.

References

Works Cited

Pew, R. and A. Mavor. 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.mediawiki.org/wiki/Help:Searching
[2] http://meta.wikimedia.org/wiki/Help:Go_button
[3] http://www.nap.edu/catalog.php?record_id=11893

Use Case 2: Other Engineers

The realization of successful complex systems requires experts from many disciplines to work together. This makes the SEBoK useful to engineers with backgrounds in biomedical, civil, electrical, chemical, civil, materials, mechanical, software, and many other engineering disciplines.

Studying the SEBoK enables engineers from disciplines other than systems engineering (SE) to

- see why good systems engineering practice must involve multiple disciplines,
- appreciate a broader view of systems beyond their specialties,
- understand how their contributions fit into the larger systems picture, and
- prepare to solve more difficult and encompassing problems.

In many cases, engineers who study systems engineering as a supplement to their area of specialization find their professional value enhanced when they put the new knowledge into practice.

Use of Topics

For engineers from non-SE backgrounds, each part of the SEBoK contributes something to the experience of learning about systems engineering.

- Part 1 provides an overview both of systems engineering and of the SEBoK itself
- Part 2 highlights the areas of systems knowledge most relevant to systems engineering, providing a foundation for the theory and practice of systems engineering as explained in Parts 3, 4 and 5
- Part 3 includes the knowledge areas of Life Cycle Models, System Definition, System Realization, and System Deployment and Use, all highly important when approaching study of SE from another discipline.
  - Also in Part 3, Systems Engineering Management includes such relevant topics as risk management, measurement, configuration management, and quality management.
- Part 4 identifies the SE activities for four kinds of engineered systems, namely products, services, enterprises, and systems of systems (SoS).
  - The primary references and glossary terms — not just the content — for a given type of system are essential reading for an engineer developing or modifying a system of that kind.
- Part 5, especially Team Capability, explains how systems engineers and other types of engineers fit into the larger picture of enabling individuals and teams to perform systems engineering activities, and into the larger picture of systems engineering organizational strategies.

- Part 6 is key for engineers from non-SE backgrounds.
  - Within Part 6, Systems Engineering and Project Management should be of interest to almost all readers, while Systems Engineering and Software Engineering and Systems Engineering and Specialty Engineering are naturally most essential for engineers in the respective disciplines.
- Part 7 illustrates how systems engineering practices, principles, and concepts are applied in real settings, and contains much universally-useful insight

Engineers may be tempted to skip over knowledge areas or topics which sound more like management than engineering stories, for example Systems Engineering Management in Part 3 or Part 5. This temptation should be resisted, because these topics are actually about how SE orchestrates the efforts of multiple disciplines, not management in the administrative sense.

Finally, the extensive lists of references throughout the SEBoK provide a basis for further readings.

Vignette: Software Engineer

Jose Wilks is an entrepreneurial software engineer who wants to learn more about systems engineering principles applied to embedded systems for advanced document identification and verification. He wants to implement best practices in developing highly secure systems for real-time image processing and forensic verification of documents. His company provides a rapid, secure and cost-effective solution for verifying the authenticity of identification, travel, and financial documents, with technology that runs on proprietary tablet computers for portable and fixed locations.

Jose is knowledgeable about computer hardware engineering, low-level interfaces between hardware and software, and the related tradeoffs in embedded devices. His company has developed research prototypes, but without the stringent security requirements for actual field usage linked to government identification databases. The few experimental units which have been sold have fared well in limited testing, but Jose wants to expand into markets for government agencies, law enforcement departments and the private sector. To make headway into those diverse markets, he will need to confront abundant new constraints and challenges.

Jose begins his study of SE by skimming the SEBoK Introduction and the Scope and Context of the SEBoK to get an overview of the SEBoK contents. As he reads, he sometimes refers to the *Software Engineering Body of Knowledge (SWEBoK)* (Bourque and Fairley 2014), which Jose already knows from his many years of experience on software projects. In the SEBoK, Jose is looking for nuggets of knowledge and pointers that can help his enterprise expand. Here are his notes:

- Part 3: Systems Engineering and Management has concepts that are new to us and that may work. Extra system-level verification and validation (V&V) gates identified in Life Cycle Models can be incorporated in company processes, and the references can help with implementation details. There is also material about system-wide procedures beyond software V&V, and about where to find testing and regulation standards used by various government entities. Together with the traditional software testing already in place, these processes could ensure conformity to the regulations and expedite the product's approval for use.
- Though the system concept is proven, the company must still convince potential buyers of the system's financial benefits while demonstrating that all security criteria are satisfied. To do that, we must understand the needs of the stakeholders better. In expressing system requirements and benefits, we need to start using the terminology of users, corporate/government purchasers, and regulatory agencies. Stakeholder Needs and Requirements is relevant here. The company needs to quantify expected return on investment (ROI) for its products.
- System Realization addresses our broader V&V concerns. We need to demonstrate the measures we are taking to boost reliability of system performance. The standard models and measures for system reliability described in the SEBoK are new to us — now staff must develop tests to quantify important attributes. We may want to model reliability and system adherence to regulations using a form of model-based systems engineering (MBSE). We can learn more about this from the references.

- Systems Engineering Management makes it clear that new configuration management (CM) and information management (IM) procedures need to be adopted for federal database controls and integrity. We can use the references in Systems Engineering Standards to learn how to define processes and develop test cases.
- Part 5: Enabling Systems Engineering makes a convincing case that having the right people for a new systems engineering culture is critical. We should probably hire a systems engineer or two to augment our engineering department expertise.
- Our application must deal with private data concerns, and Part 7: Systems Engineering Implementation Examples, the FBI Virtual Case File System Case Study could help us avoid pitfalls that have hurt others in similar situations. We can put this in context based on Security Engineering in Part 6: Related Disciplines, and then follow up with further study based on the references.

Now Jose feels that he is better prepared to adapt his processes for new system lifecycles and environments, and that he can see a clear path through the morass of agencies and regulations. His priorities are to quantify the value proposition for his technology innovations, make inroads into new markets, and strengthen his staff for the long-term enterprise.

Vignette: Mechanical Engineer

Cindy Glass is a mechanical engineer whose experience in the petroleum industry has focused on large-scale oil extraction equipment in the field. Now Cindy is tasked with helping to manage the development of new offshore oil platforms featuring robotic technology and computer networks. This calls for incorporating SE principles from day one to cope with the systems considerations, which are broader than anything in Cindy's previous experience.

Some of the drilling is to be done with remote-controlled, unmanned underwater vehicles (UUVs). Along with safety, which was always a major concern, cybersecurity now takes center stage. Hostile state actors, “hacktivists,” or others could cause havoc if they succeed in taking control of the remote vehicles or other infrastructure. Unfortunately, software system implementation is completely new to Cindy, who realizes that this entails dealing with many more engineering disciplines and dimensions of system constraints than she previously encountered.

Cindy is accustomed to implementing minor design changes in existing equipment, with automation and safety guidelines already in place. Now she is starting from scratch, with the earliest stages of the platform lifecycle. While Cindy understands tradeoffs involving mechanical sub-systems like rigs and drilling materials, she now must now broaden her system analysis to include new environmental constraints and system security.

Cindy consults the SEBoK and discovers that for her effort to understand system design with many "-ilities," System Realization is a good starting point and its references should provide the in-depth information she needs.

The project lifecycle requires pursuing several major activities concurrently:

- engineering platform sub-components
- evaluating technology opportunities
- understanding the needs of all stakeholders inside and outside the company
- progressing through increasingly detailed prototypes, working slices of software, system specifications, designs, plans, business cases, and, security and safety analyses of the platform architecture and its operations.

To understand how to manage such a project lifecycle, Cindy turns to Part 3: Systems Engineering and Management. The planning section provides detailed advice for starting out. Cindy expects to conduct her management activities on a rigorous basis, to consider the interfaces between the engineering specialties, and to produce a project plan that calls for a broad set of integrated management and technical plans.

Being new to the software development world, Cindy reads The Nature of Software and Key Points a Systems Engineer Needs to Know about Software Engineering, and consults the SWEBoK [1] for references on software engineering.

These readings show Cindy how closely systems engineering and software engineering are intertwined. For example, they remind her to include security specialists at both the software level and the systems level from the beginning.

From her initial plunge into study of the SEBoK, Cindy has gained an appreciation of the wide range of system constraints she must account for, and the many engineering disciplines she must work with as a result. She plans to consult the references in the SEBoK on each unfamiliar subject that she encounters throughout the architecting, design, development and deployment of the new platforms.

Summary

Engineers from disciplines other than systems engineering benefit from the insights about SE principles that the SEBoK provides. Studying the knowledge areas highlighted in this use case and the sources to which their references point can help such engineers become more interdisciplinary. Ultimately, they can consider broadening their work responsibilities, rendering them more valuable to their employers and society.

References

Works Cited

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.computer.org/portal/web/swebok

Use Case 3: Customers of Systems Engineering

Customers of systems engineering (SE) provide resources to SE organizations and individuals, and receive SE products and services in return. They are among the stakeholders for a system-of-interest (SoI). They and other stakeholders express needs and expectations for results that system engineers provide.

Although their main SE activity is helping to define the system, customers must take account of all life cycle aspects. The better they understand the activities that systems engineers perform, the better customers know what to request, how to request it, how much to pay for it, and how to judge the quality and value of the results of systems engineering. In short, what customers need to grasp is how systems engineers participate in the realization of engineered systems resulting in products, services, enterprises, and systems of systems (SoS).

The SEBoK assists the customers of systems engineering by providing a broad, comprehensive treatment of the concepts, principles, theory, and practice related to systems in general and SE in particular. Its references inform customers about books and articles that provide important perspectives on systems and SE.

Customers of SE include:

- sponsors of internal SE organizations
- organizations that maintain long-term customer-domain relationships with external SE organizations, and
- organizations that outsource SE functions to general-purpose SE organizations.

The two vignettes below show how the SEBoK can assist SE customers. In one, the customer of an internal, corporate SE organization leads the transition to a mobile supply chain management system. In the other, the customer of a mixture of customer-domain and other SE organizations presides over the SE of a catastrophe-response sSoS, which entails integration over multiple domains.

Use of Topics

For customers of SE, most parts of the SEBoK offer immediately relevant knowledge about SE.

Part 1:

- explains the relationship between SE, system development, and project management,
- summarizes overall trends in the rate of growth of systems interdependency, complexity, assurance levels, and pace of change, and of the evolving nature of integrated hardware-software-human systems, and
- provides pointers to other parts of the SEBoK of interest to customers.

Part 3:

- explains evolving system life cycle models and their elements, indicating which elements are SE-intensive (see Life Cycle Models),
- provides overall perspectives on customer participation in SE activity,
- identifies customer influence points on SE activity, and
- explains how customers can express their concerns in the form of needs, expectations, and requirements (see System Definition).

Part 4:

- explains how the SE function varies by class of system product, service, enterprise, and systems of systems engineering).

Part 6:

- explains how SE relates to project management, procurement and acquisition, and specialty engineering for such customer-intensive specialties as safety, security, maintainability, usability, and affordability.

Part 7:

- provides case studies and examples to illustrate how the parts have been used in similar situations, presenting successes to emulate and failures to avoid.

If there is a central theme here, it is that the quality of customer input is critical. That is because the systems engineer evaluates customer input, then uses it in formulating an approach to defining and realizing the system. Part 3 addresses this, explaining that the customer should expect the systems engineer to provide:

- a well-architected product, service, enterprise, or system of systems that meets customer needs and expectations (again, this depends on high quality input from stakeholders — see System Definition)
- a managed life cycle model from the customer need and requirements to the delivered product, service, enterprise or system of systems (see Life Cycle Models)
- verification that the system-of-interest (SoI) meets the needs and requirements of the stakeholders, and
- validation that the final result, when deployed in an operational environment, provides the value added that was desired are critical to systems engineering (see System Realization and System Deployment and Use).

Implementation Examples

Good examples provide a basis for deeper understanding. In Part 7, the SEBoK provides summaries of and references to full case studies and examples. These are linked back to the appropriate areas of the SEBoK and a matrix is provided that shows the primary areas of the SEBoK addressed by each example. Readers can use the matrix to find case studies and examples- and through these, references - that relate to their concerns.

Vignette: Mobile Supply Chain Management

Barbara Bradley is the Director of Supply Chain Management Systems for a large manufacturing company. Her main area of expertise is transportation logistics. She has led the evolution of a highly successful corporate supply chain management system based on desktop and mainframe technology, more by making incremental strategic choices than by applying formal SE.

Now, many of her suppliers and distributors adopt mobile devices and cloud services and Barbara sees that her own company must do the same. The company's status quo approach of incremental, ad hoc choices is clearly inadequate for a technology transition of this magnitude. Not only that, but the company must evolve to the new mode of operation while providing continuity of service to the supply chain stakeholders.

Barbara decides that these challenges require formal SE. As a first step, she plans to put together a Next-Generation Supply Chain Management System integrated product team (IPT). Members of the IPT will include Barbara's supply chain experts, her supply-chain success-critical stakeholders, and the corporate SE organization.

Barbara has never used the corporate SE organization before, and wants to better understand an SE organization’s overall capabilities and modes of operation. She turns to the SEBoK for answers to the questions about SE that are on her mind:

- How do we maintain continuity of service while pursuing incremental development?
  - What choices about life cycle models can make this possible?
- What is the role of the customer in defining systems of interest (SoIs)?
  - How do we provide guidance to the customer in expressing needs, concerns, and requirements?
- What is the role of the customer at early decision milestones?
  - How do we ensure that results of our interaction with the customer include well-architected products and thorough development plans, budgets, and schedules?
- What is the role of the customer in product acceptance, specifically when we verify stakeholder requirements and when we validate the final result?

Barbara seeks the answer to one question in Part 4: Applications of Systems Engineering:

- Given that a supply chain management system combines product, service, enterprise, and SoS views, how do we understand what goes into all those views, and keep the overall picture clear?

Barbara's final question is addressed in Part 6: Systems Engineering and Other Disciplines:

- How do we integrate SE and software engineering (SwE)?

Once in command of the answers to these questions, Barbara is ready to lead the IPT in analyzing, negotiating, and defining an approach that is satisfactory to all of the success-critical stakeholders. By having the IPT members read the portions of the SEBoK that she has found most valuable, Barbara begins to build a shared vision within the IPT. As the IPT defines a Next-Generation Supply Chain Management System and prepares the transition from the old system to the new, the SEBoK is an important tool and resource.

Vignette: Catastrophe-Response System of Systems

Ahmed Malik is the Information Systems Division General Manager in his country's Department of Natural Resources. The country suffers frequent wildfires that destroy crops, forests, villages, and parts of cities, and also cause problems with emergency care, crime prevention, and the water supply.

During a recent catastrophic wildfire, personnel responsible for firefighting, crime prevention, traffic control, water supply maintenance, emergency care facilities, and other key capabilities found themselves unable to communicate with each other. As a result, the Minister for Natural Resources has been tasked with improving the country's catastrophe response capabilities, and has named Ahmed as the SE customer lead for this effort.

The Minister suggests that Ahmed organize a workshop to scope the problem and explore candidate solutions to the communications problems. Ahmed invites the various actors involved in catastrophe response — medical, insurance, and news media organizations from both public and private sectors. He also invites SE organizations with SoS experience.

Ahmed has strong experience in information SE, but none in the development of SoSs. To come up to speed in his role as the SE customer lead, Ahmed turns to the SEBoK Part 3: Systems Engineering and Management. To better understand the challenges of SoS SE, he studies the SoS knowledge area in Part 4, and its references. Ahmed also schedules meetings with the leading SoS SE provider organizations, who are eager to tell him about their capabilities. Overall, Ahmed looks for both guidance and pointers to candidate solution sources in the SEBoK.

Thus prepared, Ahmed structures the workshop to address three key challenges:

- mutual understanding of organization roles, responsibilities, and authority
- summary analyses of previous catastrophe response communication gaps and needs
- candidate solution capabilities in communications, data access, geolocation services, public emergency warning systems, coordinating evacuation procedures, architectural connector approaches for improving interoperability, and sharable models for evaluating alternative solution approaches.

The workshop brings the primary organizations involved in catastrophe responses together with the most capable SoS SE provider organizations. The results of their discussions provide Ahmed and his Minister with sufficient information to prepare a phased plan, budget, and schedule for incremental development of improved catastrophe response capabilities, beginning with simple interoperability aids and analysis of architecture alternatives for performance, scalability, and feasibility of evolution from the initial simple fixes. The plan is then iterated with the key stakeholders, and converged to a common-consensus approach for achieving strong, credible early improvements and a way forward to a much more scalable and cost-effective catastrophe-response SoS.

This vignette is based on the Regional Area Crisis Response SoS (RACRS) in (Lane and Bohn 2010).

Summary

For the customers of SE, the SEBoK provides both general and specific knowledge that will help users gain important insight in relating to systems engineers. Key to this is learning about life cycles, the definition of SoIs, and how to provide guidance in expressing needs, concerns, and requirements. Further, customers need to know what to expect as a result of SE activities in the form of well-architected products, services, enterprises, or systems of systems and a managed life cycle. The results of verification of stakeholder requirements and the validation of the final result in respect to fulfilling the user needs are vital.

References

Works Cited

Lane, J. and T. Bohn. 2010. *Using SySML to Evolve Systems of Systems*. Los Angeles, CA, USA: USC CSSE Technical Report. USC-CSSE-2010-506.

Primary References

INCOSE. 2011. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.

Jamshidi, M. (ed.) 2009. *Systems of Systems Engineering - Principles and Applications*. Boca Raton, FL, USA: CRC Press.

Sage, A., and Rouse, W. (eds.) 1999. *Handbook of Systems Engineering and Management*. Hoboken, NJ, USA: John Wiley and Sons, Inc.

U.S. Department of Defense. 2008. *Systems Engineering Guide for System of Systems*, version 1.0. Arlington, VA: U.S. Department of Defense. Accessed 18 April 2013, available at: http:/ / www. acq. osd. mil/ se/ docs/ SE-Guide-for-SoS.pdf.

Additional References

P. Bourque and R.E. Fairley (eds), *Guide to the Software Engineering Body of Knowledge*, Version 3.0, IEEE Computer Society, 2014; Available at http://www.swebok.org

---

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Use Case 4: Educators and Researchers

For educators or researchers, the SEBoK should be used together with GRCSE (*Graduate Reference Curriculum for System Engineering*). The SEBoK is a guide to the knowledge that constitutes the system engineering domain, while GRCSE [1] "describes a program for a professional master's degree focused on developing student ability to perform systems engineering tasks and roles" (Pyster et al. 2012).

An educator, for purposes of this use case, is a university faculty member or a professional trainer. Educators use the SEBoK and the GRCSE to develop curricula or courses focused on systems engineering (SE) generally, on domain-centric systems engineering, or on another engineering discipline that touches on SE. The SEBoK and GRCSE are means to assure accuracy, completeness, and effective assessment at all levels, from lessons through objectives.

A researcher, for purposes of this use case, is a person actively contributing to the body of SE knowledge.

The Use of Topics

Educators can use SEBOK topics and their primary and additional references as:

- assigned readings for courses,
- supplemental references for student research, and
- content for curriculum development.

Educators can also use the concepts, perspectives, and references to develop or refine course objectives and the techniques for assessing them.

Researchers can use SEBoK topics and their primary and additional references to learn about the state of the art in the subject areas of interest, for summaries of the literature, and to look for opportunities to advance those areas by further research.

A good course or research topic should reflect multiple perspectives, which the SEBoK provides. As well, cataloging the wide diversity in accepted practices across SE is an important function of the SEBoK from the researcher's perspective.

For both educators and researchers, the fact that the SEBoK provides both primary and additional references in each topic is useful. So is the fact that the SEBoK is a wiki, which allows frequent updates to keep pace with the dynamic evolution of the systems engineering domain. See Acknowledgements and Release History.

Implementation Examples

Good examples make for good teaching. The Systems Engineering Implementation Examples in the SEBoK consist of relatively in-depth case studies and shorter examples, which are linked back to appropriate areas of the SEBoK. A matrix shows which SEBoK topics are addressed by each case study or vignette.

Each case study in the SEBoK is actually a summary of an original case found in the SE literature, and is accompanied by a reference to the full, published case study. Case study summaries or examples from the SEBoK may be incorporated in curricula.

Educator

University faculty may use the SEBoK and GRCSE to develop:

- a complete SE curriculum,
- a single course in systems engineering, either for use in an SE curriculum, or in a curriculum that belongs to some other discipline, or
- assessment criteria for curricula or courses.

Likewise, professional trainers use the SEBoK to develop training material, or to evaluate or update existing training courses.

Both faculty and trainers pursue professional development, in the form of SE study, using the SEBoK.

Vignette: Curriculum and Course Development

A university designates a faculty team to investigate the feasibility of developing a graduate degree in SE.

Results of preliminary feasibility analysis (including evaluating the market, competing degree programs, and so on) are encouraging. The faculty team then begins to design the program, by identifying:

- program constituents
- potential objectives, outcomes and entrance requirements, based on review of GRCSE
- one half of the of the curriculum content, based on review of the typical curriculum architecture (GRCSE chapter 5) and the core body of knowledge (CorBoK) (chapter 6) of GRCSE and
- the other half of the curriculum content based on review the SEBoK (Parts 2 through 7) .

According to the GRCSE, 50% of the total knowledge conveyed in a graduate program should be based on the CorBoK, to assure a common foundation among programs offered at different institutions. At the same time, restricting the CorBoK to no more than 50% encourages a healthy variety in those programs.

Once these steps are complete, the overall architecture and the content and the scope of the curriculum are defined. Now the faculty designs the courses themselves, defining in turn:

- the prerequisites for each course
- the overall course sequencing for the curriculum, based on the course prerequisites
- the objectives and goals for each course and
- the expected outcomes of each course.

Finally, the faculty is ready to develop the content for each individual course.

Defining course content is done based on topics in the SEBoK that cover the subject of the course.

Using primary and additional references as much as the topics themselves, the faculty responsible for course design define:

- the scope of the course content
- the course coverage, that is, what within the course content scope is actually taught in the course.

Given the scope and coverage, the next and final step is to develop the course material.

A professional trainer designing the training material performs the same kinds of activities. To customize the training course for a specific industry or customers, the trainer may integrate domain-specific content as well.

Researcher

Researchers use SEBoK topics and their primary and additional references to learn about the state of the art in the subject areas of topics, and to look for opportunities to advance those areas by further research.

Vignette: Software Engineering Research

William McGregor, a software engineer, wants to learn more about software intensive systems (SIS). Initially, William wants to answer the question: Do the activities and practices used to develop SIS represent special treatments of standard activities and practices?

William has already reviewed the SWEBoK and its primary references extensively for an answer to his question. In the course of his research, William learns about the SEBoK and decides to look there, too.

William finds no specific discussion of the SIS within the SEBoK. As he looks through the SEBoK, though, he realizes that there are activities throughout the system development life cycle which can be adapted or customized for the development of SIS. Accordingly, William decides to replace his original question with two new ones: (a) what best practices are applied throughout the software development life cycle and (b) how can those practices be adapted to SISs?

William now focuses on Part 3 to learn about the system development life cycle, and identify development activities and practices that he can customize for software intensive systems.

Summary

Educators use the SEBoK as a framework or a resource which helps them:

- determine what subject matter should be included in a new curriculum
- identify gaps in an existing curriculum and craft plans to address those gaps, and
- design individual courses.

The case studies and vignettes in the SEBoK may be used by educators in the classroom.

To develop curricula at the program level, educators should use the SEBoK in tandem with the GRCSE.

Researchers use the SEBoK to learn about the state of the systems engineering discipline, and to look for opportunities to advance that state by further research.

References

Works Cited

Bloom, B.S., M.D. Engelhart, E.J. Furst, W.H. Hill, and D.R. Krathwohl. 1956. *Taxonomy of Educational Objectives the Classification of Educational Goals Handbook I: Cognitive Domain.* London, UK: Longman Group Ltd.

Primary References

Pyster, A., D.H. Olwell, T.L.J. Ferris, N. Hutchison, S. Enck, J.F. Anthony, D. Henry, and A. Squires (eds). 2015. *Graduate Reference Curriculum for Systems Engineering (GRCSE™)*, version 1.1. Hoboken, NJ, USA: The Trustees of the Stevens Institute of Technology ©2015. Available at: http://www.bkcase.org/grcse-2/.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.grcse.org

Use Case 5: General Managers

General managers preside over system development projects, system acquisitions, product lines, systems of systems (SoSs), and commercial and government organizations. For general managers, the SEBoK serves as a primary information source and quick, comprehensive reference for systems engineering information.

In particular, the SEBoK helps the general manager understand:

- the boundaries and synergies among systems engineering (SE), systems development, project management (PM), and life cycle support
- how those boundaries and synergies are likely to evolve with increasing use of evolutionary development, lean and agile methods, and systems that provide purchased services as opposed to salable products
- how to best balance a mix of hardware, software, human factors, domain, and specialty-area systems engineers and
- how an organization can evolve to take advantage of the trend towards cross-discipline systems engineers.

Use of Topics

For general managers, most parts of the SEBoK offer immediately relevant knowledge about SE.

Part 1:

- explains the relationship between SE, system development, and project management
- summarizes overall trends in the nature of systems interdependency, complexity, assurance levels, and pace of change
- describes the evolving nature of integrated hardware-software-human systems and
- provides pointers to other parts of the SEBoK of interest to general managers.

Part 3:

- explains evolving system life cycle models and their elements, indicating which elements are SE-intensive (see Life Cycle Models) and
- provides overall guidance on the management of SE activity.

Part 4:

- explains how the SE function varies by class of system product, service, enterprise, and systems of systems engineering).

Part 5:

- explains SE governance and competence development.

Part 6:

- explains how SE relates to software engineering, project management, industrial engineering, procurement and acquisition, and specialty engineering for such specialties as safety, security, maintainability, and usability.

Part 7:

- provides case studies and vignettes to illustrate how the parts have been used in similar situations in successes to emulate and failures to avoid.

Vignette: Emerging Nation Satellite System

Tom Lee is General Manager for Telecommunications in a ministry of a large emerging nation. The government does not have much existing capability for developing capital-intensive infrastructure projects. The government decides to use a major investment in technology as a vehicle to develop national enterprise capabilities.

To accomplish this, the minister assigns Tom to lead a project to develop a national satellite system for telecommunications and earth resources observation. Tom understands that this is a very complex system, and decides to do some background research. During this research, Tom discovers the SEBoK and decides that is may be a useful resource.

Tom first reads:

- Part 1 for an overview and pointers to relevant sections of Parts 3 through 6,
- portions of Part 3, Part 4, Part 5, and Part 6 to learn about the life cycle, nature, scope, and management aspects of enterprise SE,
- the successful satellite system case studies in Part 7 (Global Positioning System, Miniature Seeker Technology Integration spacecraft) for approaches to emulate, and
- the satellite system case study in Part 7 which describes development and integration problems (Hubble Space Telescope) for pitfalls to avoid.

Tom continues by carefully reading Part 5. He realizes that he must develop simultaneously individuals, teams, and the enterprise. The knowledge areas (KAs) from Part 5 give useful background. For this project, Tom enlists both a proven multi-national satellite SE company and some of his brightest aerospace systems engineers. Tom expects his local systems engineers to learn from the SE company, and he plans to use them as the core group of the national satellite system as it ultimately develops and operates.

He realizes that correct problem definition and requirements setting will be critical first steps. He carefully reads the Concept Definition and System Definition KAs. As his team develops the Stakeholder Needs and Requirements and the System Requirements, he makes sure they follow good practices as listed in the SEBoK. Once architectural designs have been proposed and approved, he requires his team to perform cost-benefit tradeoff analyses of alternative solutions.

Thus prepared, Tom is confident that he can formulate and execute a successful approach.

Vignette: Commercial Safety Equipment Company

Maria Moreno is General Manager at Safety First Equipment Company, specialists in hardware-intensive safety equipment. Maria’s background is in electromechanical systems. Safety First is highly successful, but beginning to lose market share to competitors who offer software-intensive capabilities and user amenities.

Maria is preparing an initiative to make Safety First into a leading software-intensive safety equipment provider. She decides to make the SEBoK a primary resource for gathering concepts and insights for the initiative. She begins by skimming through all of Parts 1 through 6, both to become familiar with the SEBoK itself and to start organizing her thoughts on SE.

Now Maria is ready to focus on subjects of prime importance to her task. Here are those subjects, listed with the places in the SEBoK where she find information about them.

In Systems Engineering and Software Engineering in Part 6:

- the nature of software
- differences between hardware and software architectures and practices and

- key aspects of managing software teams.

In the article Human Systems Integration in the Systems Engineering and Specialty Engineering knowledge area, also in Part 6:

- the SE of user amenities.

In the Next Generation Medical Infusion Pump Case Study in Part 7:

- the software aspects of safety practices, such as software fault tree analysis and failure modes and effects analysis and
- overall approaches for concurrent engineering of the hardware, software, and human factors aspects of safety-critical equipment.

In the Medical Radiation Case Study in Part 7:

- hardware-software pitfalls to avoid in safety-critical equipment.

Maria chose the last two items from among the case studies in Part 7 because being safety-critical, they contain lessons directly applicable to her initiative at Safety First.

With this framework of concepts and practical information in place, Maria begins assembling a core team of Safety First systems engineers, complemented by external experts in software and human factors engineering. Maria wants the team to begin by developing a shared vision. To that end, she asks them to read the portions of the SEBoK that she has found most valuable in assessing the challenges of transitioning Safety First into a leading software-intensive, user-friendly safety equipment provider.

Summary

For the general manager whose organization includes systems engineers, the relationship between SE, systems development, project management, and life cycle support is a central concern. The SEBoK provides insights and guidance about this and other aspects of SE principle and practice, and explains the role of SE in a variety of management challenge areas and application domains.

The SEBoK complements the general management guidance available in sources such as the *PMBOK® Guide* (PMI 2013).

References

Works Cited

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

P. Bourque and R.E. Fairley (eds), *Guide to the Software Engineering Body of Knowledge*, Version 3.0, IEEE Computer Society, 2014; Available at http://www.swebok.org

Booher, H. 2003. *Handbook of Human-Systems Integration*. New York, NY, USA: John Wiley & Sons Inc.

Hooks, I.F. and K. Farry. 2000. *Customer Centered Products: Creating Successful Products Through Smart Requirements Management*. New York NY, USA: AMACON/American Management Association.

Pew, R. and A. Mavor. 2007. *Human-System Integration in the System Development Process*. Washington, DC, USA: The National Academies Press.

< Previous Article | Parent Article | Next Article(Part 2) >

SEBoK v. 1.9.1, released 16 October 2018

Part 2: Foundations of Systems Engineering

Foundations of Systems Engineering

Part 2 of the Guide to the SE Body of Knowledge (SEBoK) is a guide to foundational knowledge which is relevant or useful to systems engineering (SE).

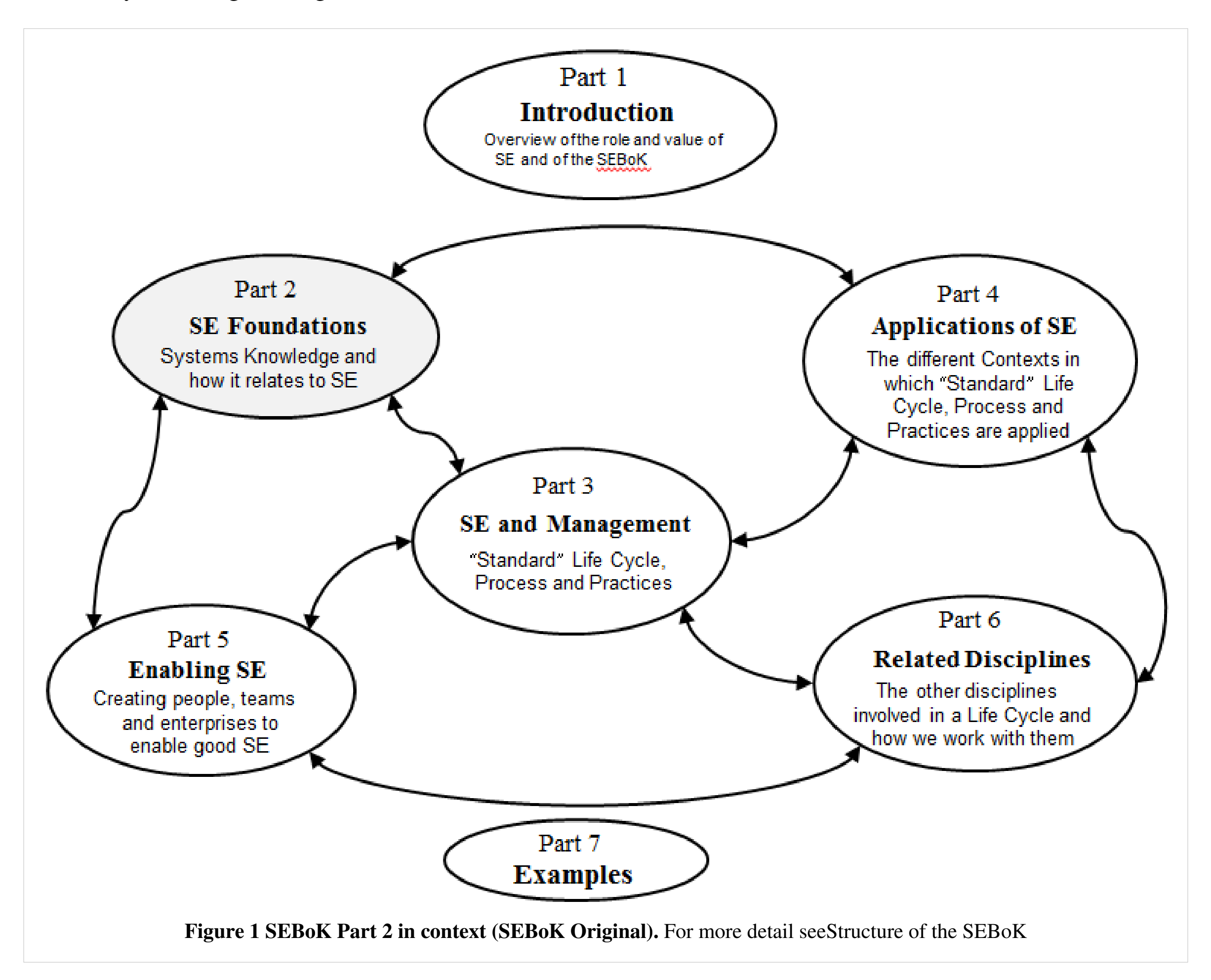


Figure 1 SEBoK Part 2 in context (SEBoK Original). For more detail seeStructure of the SEBoK

This knowledge is included in the SEBoK firstly to help systems engineers benefit from an understanding of the foundations of their discipline, and to provide them with access to some of the theories and practices of systems science and other fields of systems practice. Including this wider integrative systems science context in the SEBoK should also help to make SE knowledge more accessible to a wider audience outside of its traditional domains.

Knowledge Areas in Part 2

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 2 contains the following KAs:

- Systems Fundamentals
- Systems Science
- Systems Thinking
- Representing Systems with Models
- Systems Approach Applied to Engineered Systems

Introduction

Most systems engineers are practitioners, applying processes and methods that have been developed and evolved over decades. SE is a pragmatic approach, inherently interdisciplinary, yet specialized. Systems engineers usually work within a specific domain, using processes and methods that are tailored to their domain's unique problems, constraints, risks and opportunities. These processes and methods have evolved to capture domain experts' knowledge regarding the best approach to applying SE the particular domain.

Specific domains in which systems approaches are used and adapted include:

- Technology products, integrating multiple engineering disciplines
- Information-rich systems, e.g. command & control, air traffic management etc.
- Platforms, e.g. aircraft, civil airliners, cars, trains, etc.
- Organizational and enterprise systems, which may be focused on delivering service or capability
- Civil engineering/infrastructure systems, e.g. roads networks, bridges, builds, communications networks, etc.

The specific skill-sets for each domain, and the kinds and scales of system it considers, may be quite different. However, there are certain underlying unifying systems principles that can improve the effectiveness of the systems approach in any domain. In particular, shared knowledge of systems principles and terminology will enable communication and improve system engineers' ability to integrate complex systems that span traditional domain boundaries (Sillitto 2012). This integrated approach is increasingly needed to solve today's complex system challenges, but as these different communities come together they may find that assumptions underpinning their world-views are not shared.

To bridge the gap between different domains and communities of practice, it is important to first establish a well-grounded definition of the "intellectual foundations of systems engineering", as well as a common language to describe the relevant concepts and paradigms. An integrated systems approach for solving complex problems needs to combine elements of systems theories and systems approaches to practice. This may range from the technical-systems focus that has been dominant in systems engineering to the learning-systems focus of social systems intervention. An integrated systems approach needs to provide a framework and language that allow different communities, with highly divergent world-views and skill sets, to work together for a common purpose.

The Systems Praxis Framework

The term **"systems praxis"** refers to the entire intellectual and practical endeavor for creating holistic solutions to today's complex system challenges. Praxis is defined as "translating an idea into action" (Wordnet 2012) and suggests that the best holistic approach to a given complex challenge may require integrating appropriate theory and appropriate practice from a wide variety of sources. Systems praxis requires many communities to work together. To work together we must first communicate; and to communicate, we must first connect.

A framework for unifying systems praxis was developed by members of International Council on Systems Engineering (INCOSE) and International Society for the System Sciences (ISSS) (International Federation for Systems Research (IFSR) 2012)) as the first step towards a "common language for systems praxis". This **Systems**

Praxis Framework is included here because it represents current thinking on the foundations and common language of systems engineering, making the concepts and principles of systems thinking and practice accessible to anyone applying a systems approach to engineered system problems. This framework and thinking have been used to help organize the guide to systems knowledge in the SEBoK.

The diagram below shows the flows and interconnections among elements of a "knowledge ecosystem" of systems theory and practice.

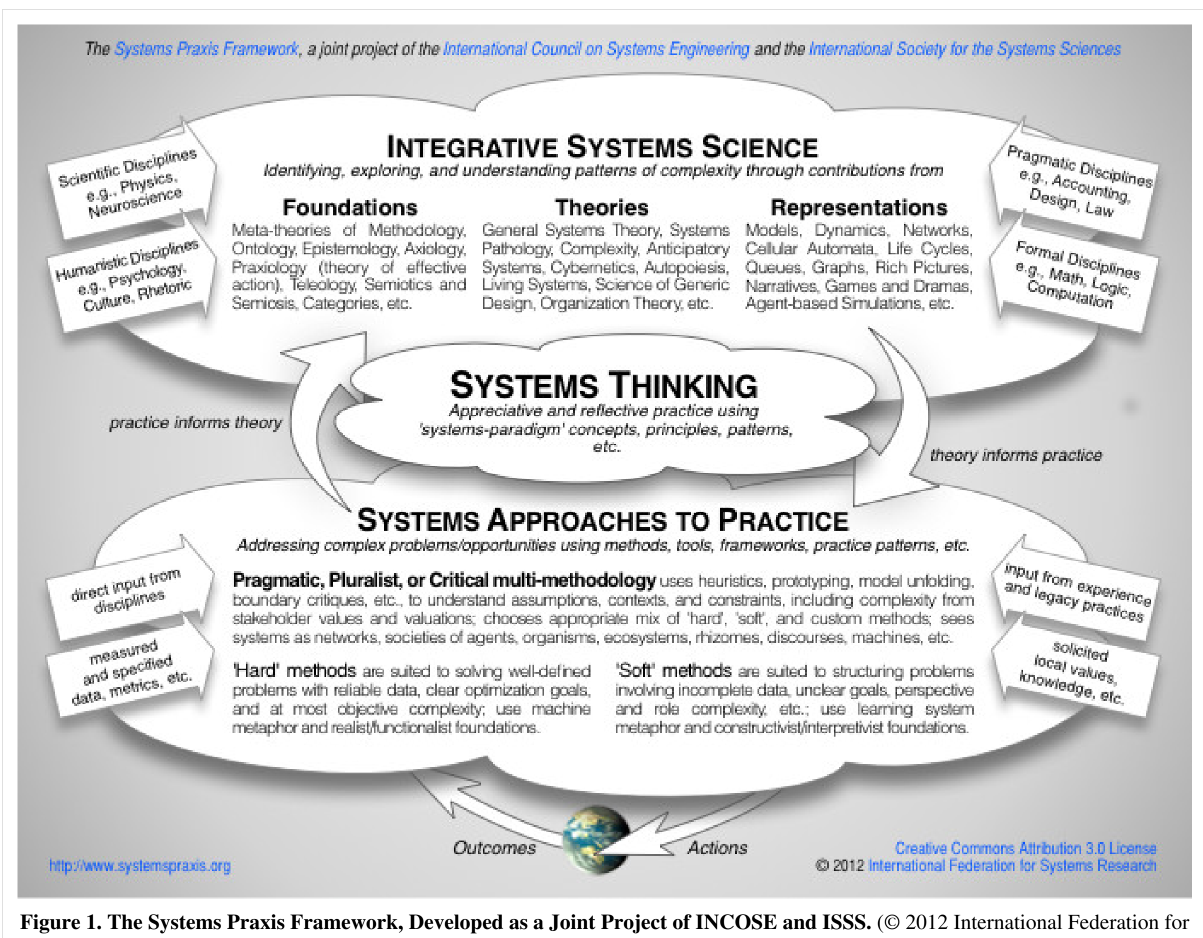


Figure 1. The Systems Praxis Framework, Developed as a Joint Project of INCOSE and ISSS. (© 2012 International Federation for Systems Research) Released under Creative Commons Attribution 3.0 License. Source is available at http://systemspraxis.org/framework.pdf.

In this framework, the following elements are connected:

Systems Thinking is the core integrative element of the framework. It binds the foundations, theories and representations of systems science together with the hard, soft and pragmatic approaches of systems practice. In systems praxis, as in any practical discipline underpinned by science, there is constant interplay between theories and practice, with theory informing practice and outcomes from practice informing theory. Systems thinking is the ongoing activity of assessing and appreciating the system context, and guiding appropriate adaptation, throughout the praxis cycle.

Integrative Systems Science has a very wide scope and is grouped into three broad areas:

- **Foundations**, which help to organize knowledge and promote learning and discovery including: meta-theories of methodology, ontology, epistemology, axiology, praxiology (theory of effective action), teleology, semiotics & semiosis, category theory, etc.
- **Theories** pertaining to systems are abstracted from domains and specialties, so as to be universally applicable: general system theory, systems pathology, complexity, anticipatory systems, cybernetics, autopoiesis, living systems, science of generic design, organization theory, etc.

- **Representations** and corresponding theories describe, explore, analyze, and make predictions about systems and their wider contexts, whether in terms of models, dynamics, networks, cellular automata, life cycles, queues, graphs, rich pictures, narratives, games and dramas, agent-based simulations, etc.

Systems Approaches to Practice aim to act on real world experiences to produce desired outcomes without adverse, unintended consequences; ergo, practice needs to draw on the wide range of knowledge appropriate to the system-of-interest and its wider context. No one branch of systems science or practice provides a satisfactory explanation for all aspects of a typical system “problematique”; therefore, a more pragmatic approach is needed. Traditional systems approaches are often described to be either hard or soft:

- **Hard** approaches are suited to solving well-defined problems with reliable data and clear goals, using analytical methods and quantitative techniques. Strongly influenced by “machine” metaphors, they focus on technical systems, objective complexity, and optimization to achieve desired combinations of emergent properties. They are based on “realist” and “functionalist” foundations and worldview.
- **Soft** approaches are suited to structuring problems involving incomplete data, unclear goals, and open inquiries, using a “learning system” metaphor, focus on communication, intersubjective complexity, interpretations and roles, and draw on subjective and “humanist” philosophies with constructivist and interpretivist foundations.

Pragmatic (pluralist or critical) approaches judiciously select an appropriate set of tools and patterns that will give sufficient and appropriate insights to manage the issue at hand, by applying multiple methodologies drawn from different foundations as appropriate to the situation. Heuristics, boundary critiques, model unfolding, etc, enable the understanding of assumptions, contexts, and constraints, including complexity due to different stakeholders’ values and valuations. An appropriate mix of “hard”, “soft”, and custom methods draws on both systems and domain-specific traditions. Systems may be viewed as networks, societies of agents, organisms, ecosystems, rhizomes, discourses, machines, etc.

The set of “clouds” that collectively represents systems praxis is part of a wider ecosystem of knowledge, learning, and action. Successful integration with this wider ecosystem is the key to success with real world systems. Systems science is augmented by “hard” scientific disciplines, such as physics and neuroscience, and by formal disciplines, such as mathematics, logic and computation. It is both enhanced by, and used in, humanistic disciplines, such as psychology, culture, and rhetoric, and pragmatic disciplines, such as accounting, design, and law. Systems practice depends on measured data and specified metrics relevant to the problem situation and domain, the solicitation of local values and knowledge, and the pragmatic integration of experience, legacy practices, and discipline knowledge.

In summary, **Integrative Systems Science** allows us to identify, explore, and understand patterns of complexity through contributions from the foundations, theories, and representations of systems science and other disciplines relevant to the “problematique”. **Systems Approaches to Practice** address complex problems and opportunities using methods, tools, frameworks, patterns, etc., drawn from the knowledge of integrative systems science, while the observation of the results of systems practice enhances the body of theory. **Systems Thinking** binds the two together through appreciative and reflective practice using systems concepts, principles, patterns, etc.

Scope of Part 2

Part 2 of the SEBoK contains a guide to knowledge about systems, which is relevant to a better understanding of SE. It does not try to capture all of this systems knowledge here; rather, it provides an overview of a number of key aspects of systems theory and practice especially relevant to SE.

The organization of knowledge in Part 2 is based around the Praxis Framework discussed above (IFSR 2012). The need to develop a clear guide to the underpinning knowledge of SE is one of the motivations behind the praxis framework. It is expected that the coverage of systems knowledge will be significantly increased in future versions of the SEBoK as this work progresses.

The following diagram summarizes the way in which the knowledge in SEBoK Part 2 is organized.

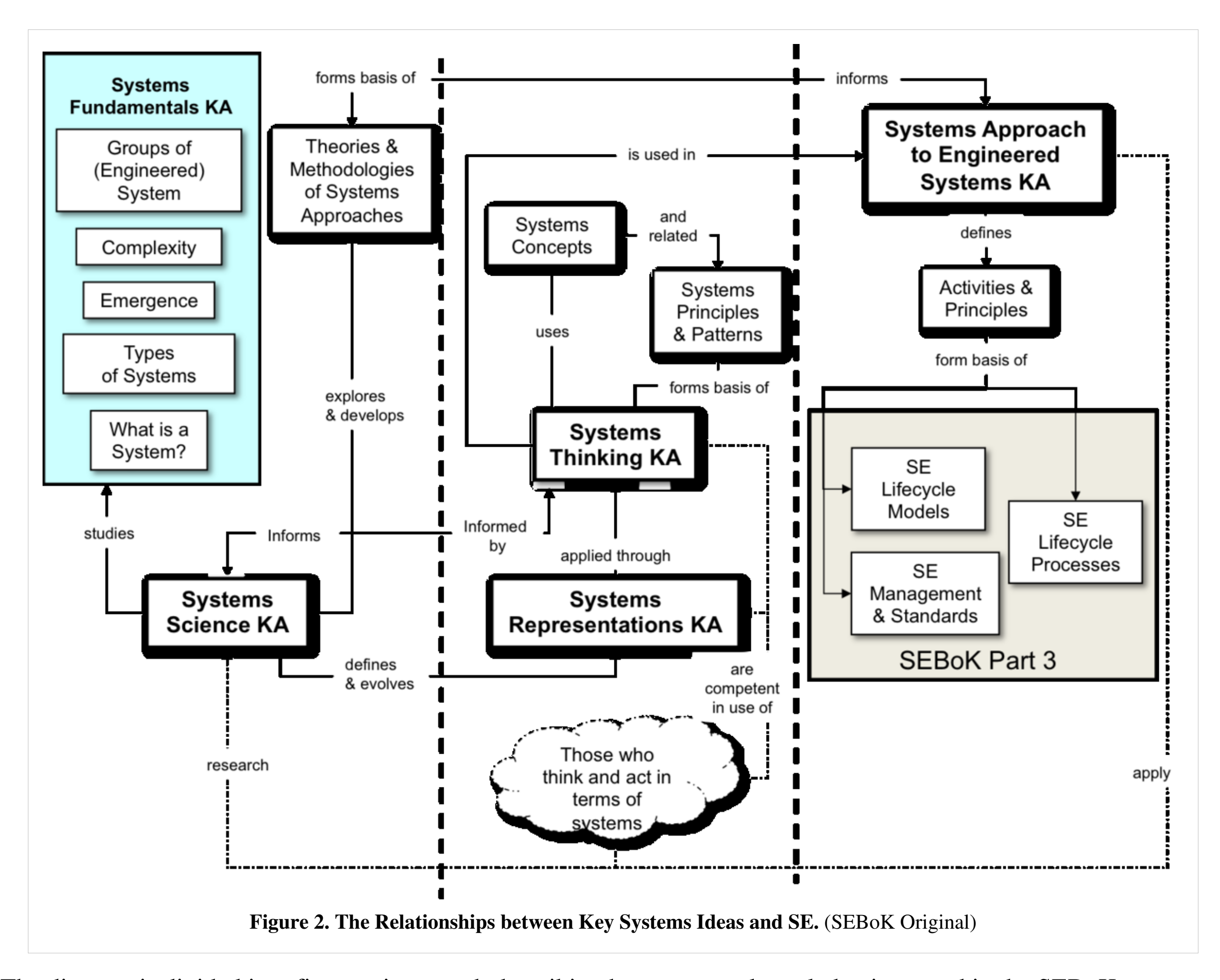


Figure 2. The Relationships between Key Systems Ideas and SE. (SEBoK Original)

The diagram is divided into five sections, each describing how systems knowledge is treated in the SEBoK.

1. The Systems Fundamentals Knowledge Area considers the question “What is a System?” It explores the wide range of system definitions and considers open systems, system types, groupings of systems, complexity, and emergence. All of these ideas are particularly relevant to engineered systems and to the groupings of such systems associated with the systems approach applied to engineered systems (i.e. product system, service system, enterprise system and system of systems).
2. The Systems Science Knowledge Area presents some influential movements in systems science, including the chronological development of systems knowledge and underlying theories behind some of the approaches taken in applying systems science to real problems.
3. The Systems Thinking Knowledge Area describes key concepts, principles and patterns shared across systems research and practice.
4. The Representing Systems with Models Knowledge Area considers the key role that abstract models play in both the development of system theories and the application of systems approaches.
5. The Systems Approach Applied to Engineered Systems Knowledge Area defines a structured approach to problem/opportunity discovery, exploration, and resolution, that can be applied to all engineered systems. The approach is based on systems thinking and utilizes appropriate elements of system approaches and representations. This KA provides principles that map directly to SE practice.

Systems thinking is a fundamental paradigm describing a way of looking at the world. People who think and act in a systems way are essential to the success of both the research and practice of system disciplines. In particular, individuals who have an awareness and/or active involvements in both research and practice of system disciplines are needed to help integrate these closely related activities.

The knowledge presented in this part of the SEBoK has been organized into these areas to facilitate understanding; the intention is to present a rounded picture of research and practice based on system knowledge. These knowledge areas should be seen together as a “system of ideas” for connecting research, understanding, and practice, based on system knowledge which underpins a wide range of scientific, management, and engineering disciplines and applies to all types of domains.

References

Works Cited

IFSR. 2012. *The Systems Praxis Framework, developed as a joint project of INCOSE and ISSS*. Vienna, Austria: International Federation for Systems Research (IFSR). Source is available at http://systemspraxis.org/framework.pdf.

Sillitto, H G, 2012. "Integrating Systems Science, Systems Thinking, and Systems Engineering: understanding the differences and exploiting the synergies", Proceedings of the 22nd INCOSE International Symposium, 9-12 July, 2012, Rome, Italy.

Wordnet. 2012. “Praxis.” Accessed 4/16/2013 at http://wordnetweb.princeton.edu/perl/webwn?s=praxis&sub=Search+WordNet&o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&h=

Primary References

Bertalanffy, L., von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York, NY, USA: Braziller.

Checkland, P. B. 1999. *Systems Thinking, Systems Practice.* Chichester, UK: John Wiley & Sons.

Additional References

Blanchard, B., and Fabrycky, W. 2010. *Systems Engineering and Analysis*, (5th edition). Saddle River, NJ, USA: Prentice Hall.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College, UK.

Martin J, Bendz J, Chroust G, Hybertson D, Lawson H, Martin R, Sillitto H, Singer J, Singer M, Takaku T. “Towards a Common Language for Systems Praxis”, proceedings of the 23rd INCOSE International Symposium, Philadelphia, June 2013.

MITRE Corporation. 2011. *Systems Engineering Guide: Comprehensive Viewpoint.*. Accessed 20 November 2014 at MITRE http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/comprehensive_viewpoint/

MITRE Corporation. 2011. *Systems Engineering Guide: Systems Thinking.*. Accessed 20 November 2014 at MITRE http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/comprehensive_viewpoint/systems_thinking.html

Senge, P. M. 1990. *The Fifth Discipline: The Art & Practice of the Learning Organization.* New York, NY: Doubleday Business.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 12 October 2018

Knowledge Area: Systems Fundamentals

Systems Fundamentals

This knowledge area (KA) provides a guide to some of the most important knowledge about a system, which forms part of systems thinking and acts as a foundation for the related worlds of integrative systems science and systems approaches to practice.

This is part of the wider systems knowledge, which can help to provide a common language and intellectual foundation and make practical systems concepts, principles, patterns and tools accessible to systems engineering (SE) as discussed in Part 2: Foundations of Systems Engineering.

Topics

Each part of the SEBoK is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- What is a System?
- Types of Systems
- Groupings of Systems
- Complexity
- Emergence

Introduction

The word *system* is used in many areas of human activity and at many levels. But what do systems researchers and practitioners mean when they use the word *system*? Is there some part of that meaning common to all applications? The following diagram summarizes the ways in which this question is explored in this KA.

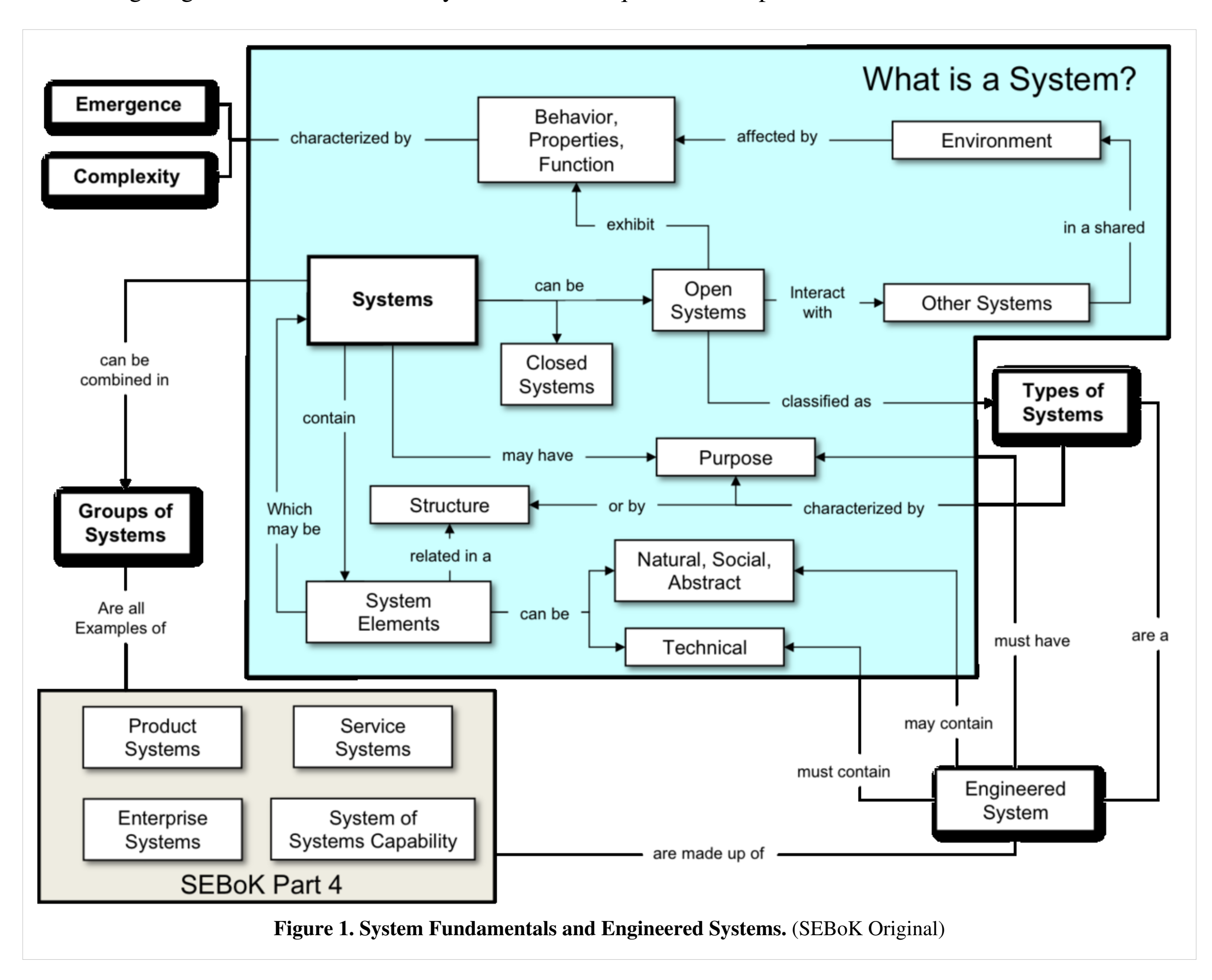


Figure 1. System Fundamentals and Engineered Systems. (SEBoK Original)

The concepts of open system and closed system are explored. Open systems, described by a set of elements and relationships, are used to describe many real world phenomena. Closed systems have no interactions with their environment. Two particular aspects of systems, complexity and emergence, are described in this KA. Between them, these two concepts represent many of the challenges which drive the need for systems thinking and an appreciation of systems science in SE.

Some systems classifications, characterized by type of element or by purpose, are presented.

Within the SEBoK an engineered system is defined as encompassing combinations of technology and people in the context of natural, social, business, public or political environments, created, used and sustained for an identified purpose. The application of the Systems Approach Applied to Engineered Systems requires the ability to position problems or opportunities in the wider system containing them, to create or change a specific engineered system-of-interest, and to understand and deal with the consequences of these changes in appropriate wider systems. The concept of a system context allows all of the system elements and relationships needed to support this to be identified.

The discussions of engineered system contexts includes the general idea of groups of systems to help deal with situations in which the elements of an engineered system are themselves independent engineered systems. To help provide a focus for the discussions of how SE is applied to real world problems, four engineered system contexts are introduced in the KA:

1. Product System (glossary) context
2. Service System (glossary) context
3. Enterprise System (glossary) context
4. System of Systems (SoS) (glossary) capability context

The details of how SE is applied to each of these contexts are described in Part 4: Applications of Systems Engineering.

References

Works Cited

None.

Primary References

Bertalanffy, L., von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York, NY, USA: Braziller.

Magee, C. L., O.L. de Weck. 2004. "Complex System Classification." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June 2004, Toulouse, France.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press.

Sheard, S.A. and A. Mostashari. 2009. "Principles of Complex Systems for Systems Engineering". *Systems Engineering*, 12(4): 295-311.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering*, 12(1): 13-38.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

What is a System?

This article forms part of the Systems Fundamentals knowledge area (KA). It provides various perspectives on systems, including definitions, scope, and context. The basic definitions in this article are further expanded and discussed in the articles Types of Systems and What is Systems Thinking?.

This article provides a guide to some of the basic concepts of systems developed by systems science and discusses how these relate to the definitions to be found in systems engineering (SE) literature. The concept of an engineered system is introduced as the system context of critical relevance to SE.

A Basic View of Systems

The most basic ideas of a system whole can be found in both Western and Eastern philosophy. Many philosophers have considered notions of holism; that ideas, people or things must be considered in relation to the things around them to be fully understood (M’Pherson 1974).

One influential systems science definition of a system comes from general system theory (GST):

> "A System is a set of elements in interaction." (Bertalanffy 1968)

The parts of a system may be conceptual organizations of ideas in symbolic form or real objects. GST considers **abstract systems** to contain only conceptual elements and **concrete systems** to contain at least two elements that are real objects, e.g. people, information, software and physical artifacts, etc.

Similar ideas of wholeness can be found in systems engineering literature. For example:

> *We believe that the essence of a system is 'togetherness', the drawing together of various parts and the relationships they form in order to produce a new whole…* (Boardman and Sauser 2008).

The cohesive interactions between a set of parts, (Hitchins 2009, 59-63), allow us to identify a system boundary and defined what membership of the system means. For **closed systems** all aspects of the system exist within this boundary. This idea is useful for abstract systems and for some theoretical system descriptions.

The boundary of an **open systems** (glossary) defines those elements and relationships which can be considered part of the system and those which describe the interactions across the boundary between system elements and elements in the environment (glossary). The relationships between the various elements of an open system can be understood as a combination of the system structure and behavior. The structure of a system describes a set of system elements and the allowable relationships between them. System behavior refers to the effects or outcomes produced when an instance of the system interacts with its environment. An allowable configuration of the relationships between elements is referred to as a system state. A stable system is one which returns to its original, or another, state following a disturbance in the environment.

The identification of a system and its boundary is ultimately the choice of the observer. This may be through observation and classification of sets of elements as systems, through an abstract conceptualisation of one or more possible boundaries and relationships in a given situation, or a mixture of both concrete and conceptual thinking. This underlines the fact that any particular identification of a system is a human construct used to help make better sense of a set of things and to share that understanding with others if needed.

Many natural, social and man made things can be better understood by describing them as open systems. One of the reasons we find the idea of systems useful is that it is possible to identify shared concepts which apply to systems, and give useful insights into them, independently of the kinds of elements they are made up of. The ideas of structure, behavior and state are examples of such concepts. The identification of these shared system ideas, is the basis for Systems Thinking and their use in developing theories and approaches in a wide range of fields of study the system sciences.

Systems Engineering (SE), and a number of other Related Disciplines also use systems concepts, patterns and models in the creation of useful outcomes or things. The concept of a network of open systems created, sustained and used to achieve a purpose within one or more environments is a powerful model that can be used to understand many complex real world situations and provide a basis for effective problem solving.

Systems and System Elements

Many of the original ideas upon which GST, and other branches of system study, is based come from the study of systems in the biological and social sciences. Many natural and social systems are formed through the inherent cohesion between elements. Once formed, they will tend to stay in this structure, as well as combine and evolve further into more complex stable states to exploit this cohesion in order to sustain themselves in the face of threats or environmental pressures. Such complex systems may exhibit specialization of elements, with elements taking on roles which contribute to the system purpose. Such roles might include self-regulation or control. Natural and social systems can be understood through an understanding of this wholeness, cohesion and specialization. They can also be guided towards the development of behaviors which not only enhance their basic survival, but also fulfill other goals or benefit to them or the systems around them. *The Architecture of Complexity* (Simon 1962) has shown that systems which evolve via a series of stable "hierarchical intermediate forms" will be more successful and adapt more quickly to environmental change.

Many man-made systems are designed as networks and hierarchies of related system elements to achieve similar resilience and desired behavior. Thus, it is often true that the elements of a system can themselves be considered as open systems. It can be useful to consider collections of related elements as both a system and a part of one or more other systems. A "holon" or System Element (glossary) was defined by Koestler as something which exists simultaneously a whole and as a part (Koestler 1967). Networks of related open systems at different levels of detail are the starting point for many applications of SE. The Systems Thinking knowledge area includes an overview of concepts, principles and patterns related to these types of system. There are two aspects of these systems which are of particular interest. Firstly, *whole entities exhibit emergence, properties which are meaningful only when attributed to the whole, not to its parts* (Checkland 1999). At some point, the nature of the relationships between elements within and across boundaries in a hierarchy of systems may lead to complex behavior which is difficult to understand or predict. Both emergence and complexity can often best be dealt with not by looking for more detail, but by considering the wider open system relationships.

Bertalanffy (1968) divided open systems into nine real world types ranging from static structures and control mechanisms to socio-cultural systems. Other similar classification systems are discussed in the article Types of Systems. The following is a simple classification of system elements:

- Natural system elements, objects or concepts which exist outside of any practical human control. Examples: the real number system, the solar system, planetary atmosphere circulation systems.
- Social system elements, either abstract human types or social constructs, or concrete individuals or social groups.
- Technological System elements, man-made artifacts or constructs; including physical hardware, software and information.

While the above distinctions can be made as an abstract classification, in reality, these are not hard and fast boundaries between these types of systems: e.g., social systems are operated by, developed by, and also contain natural systems and social systems depend on technical systems to fully realize their purpose. Systems which contain technical and either human or natural elements, are often called socio-technical systems. The behavior of such systems is determined both by the nature of the technical elements and by their ability to integrate with or deal with the variability of the natural and social systems around them.

System-of-Interest and Systems Context

Considering the relationships between a number of relevant system elements helps to fully explore problems and solutions. A given element may be included in several system views. However, we may also want to focus on a particular system as the primary focus of output of these activities. The idea of a System Context is used to define a System of Interest (SoI) and to identify the important relationships between its internal system elements, the wider system elements it works directly with, and the environmental conditions which influence it in some way. Context is a very important foundation of SE, since it combines the broad consideration of many related system elements with a focus on a particular socio-technical system deliverable.

The term socio-technical system is used by many in the systems community and may have meanings outside of that relevant to SE. Hence, we will define an Engineered System (glossary) as a system context in which a socio-technical system forms the primary focus across the whole of its Life Cycle (glossary), from initial problem formulation through to its final safe removal from use (INCOSE 2012). Engineered systems can be deliberately created to take advantage of system properties such as holism and stability, but must also consider system challenges such as complexity and emergence.

Systems engineering literature, standards and guides generally refer to their system-of-interest (SoI) as “the system” and their definitions of “a system” tend to characterize socio-technical (SoI) with a defined purpose, e.g.

- “A system is a value-delivering object” (Dori 2002).
- “A system is an array of components designed to accomplish a particular objective according to plan” (Johnson, Kast, and Rosenzweig 1963).
- “A system is defined as a set of concepts and/or elements used to satisfy a need or requirement" (Miles 1973).

The International Council on Systems Engineering Handbook (INCOSE) (INCOSE 2012) generalizes this idea, defining system as “an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements." While this definition generally covers the SoI, we may also need to consider natural and social systems within the environment to fully understand an engineered system context. In SE and its related disciplines an engineered SoI is created and used by people for a purpose and may need to be considered across the whole of its life. While this is an important part of SE we can also consider the problem situation in which the engineered system sits, the social systems which developed, sustained and used them, and the commercial or public enterprises in which these all sit as systems (Martin 2004).

Hence, while many SE authors talk about systems and systems ideas they are often based on a particular world view which related to engineered systems and is focused on the creation of one or more systems of interest. It would also be useful to take a broader view of the context in which these SoI sit, and to consider through life relationships as part of that context. To help promote this the SEBoK will try to be more precise with its use of the word system, and will consider a wide set of system principles, pattern and models.

References

Works Cited

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York: Braziller.

Boardman, J. and B. Sauser. 2008. *Systems Thinking: Coping with 21st Century Problems.* Boca Raton, FL, USA: Taylor & Francis.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: Wiley and Sons, Inc.

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. Verlag, Berlin, Heidelberg, New York: Springer.

Hitchins, D. 2009. "What Are the General Principles Applicable to Systems?" INCOSE *Insight*, 12(4): 59-63.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Johnson, R.A., F.W. Kast, and J.E. Rosenzweig. 1963. *The Theory and Management of Systems.* New York, NY, USA: McGraw-Hill Book Company.

Koestler, A. 1990. *The Ghost in the Machine,* 1990 reprint ed. Penguin Group.

Martin, J, 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems". Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June, 2004, Toulouse, France.

Miles, R.F. (ed). 1973. *System Concepts.* New York, NY, USA: Wiley and Sons, Inc.

M'Pherson, P.K. 1974. "A perspective on systems science and systems philosophy". *Futures*. 6(3):219-39.

Simon, H.A. 1962. "The Architecture of Complexity." *Proceedings of the American Philosophical Society.* 106(6) (Dec. 12, 1962): 467-482.

Primary References

Bertalanffy, L., von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York, NY, USA: Braziller.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Additional References

Hybertson, Duane. 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL, USA: CRC Press.

Hubka, Vladimir, and W. E. Eder. 1988. *Theory of Technical Systems: A Total Concept Theory for Engineering Design*. Berlin: Springer-Verlag.

Laszlo, E., ed. 1972. *The Relevance of General Systems Theory: Papers Presented to Ludwig von Bertalanffy on His Seventieth Birthday.* New York, NY, USA: George Brazillier.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 12 October 2018

Types of Systems

This article forms part of the Systems Fundamentals knowledge area (KA). It provides various perspectives on system classifications and types of systems, expanded from the definitions presented in What is a System?.

The modern world has numerous kinds of systems that influence daily life. Some examples include transport systems; solar systems; telephone systems; the Dewey Decimal System; weapons systems; ecological systems; space systems; etc. Indeed, it seems there is almost no end to the use of the word "system" in today's society.

This article considers the different classification systems which some Systems Science (glossary) authors have proposed in an attempt to extract some general principles from these multiple occurrences. These classification schemes look at either the kinds of elements from which the system is composed or its reason for existing.

The idea of an engineered system (glossary) is expanded. Four specific types of engineered system context are generally recognized in systems engineering: product system, service system, enterprise system and system of systems capability.

System Classification

A taxonomy is "a classification into ordered categories" (Dictionary.com 2011). Taxonomies are useful ways of organizing large numbers of individual items so their similarities and differences are apparent. No single standard classification system exists, though several attempts have been made to produce a useful classification taxonomy. Kenneth Boulding (Boulding 1956), one of the founding fathers of general system theory, developed a systems classification which has been the starting point for much of the subsequent work. He classifies systems into nine types:

1. Structures (Bridges)
2. Clock works (Solar system)
3. Controls (Thermostat)
4. Open (Biological cells)
5. Lower organisms (Plants)
6. Animals (Birds)
7. Man (Humans)
8. Social (Families)
9. Transcendental (God)

Bertalanffy (1968) divided systems into nine types, including control mechanisms, socio-cultural systems, open systems, and static structures. Miller (Miller 1986) offered cells, organization, and society among his eight nested hierarchical living systems levels, with twenty critical subsystems at each level.

Peter Checkland (Checkland 1999, 111) divides systems into five classes: natural systems, designed physical systems, designed abstract systems, human activity systems and transcendental systems. The first two classes are self-explanatory.

- **Designed abstract systems** – These systems do not contain any physical artifacts but are designed by humans to serve some explanatory purpose.
- **Human activity systems** – These systems are observable in the world of innumerable sets of human activities that are more or less consciously ordered in wholes as a result of some underlying purpose or mission. At one extreme is a system consisting of a human wielding a hammer. At the other extreme lies international political

systems.

- **Transcendental systems** – These are systems that go beyond the aforementioned four systems classes, and are considered to be systems beyond knowledge.

Checkland refers to these five systems as comprising a “systems map of the universe”. Other, similar categorizations of system types can be found in (Aslaksen 1996), (Blanchard 2005) and (Giachetti 2009).

These approaches also highlight some of the subsequent issues with these kinds of classification. Boulding implies that physical structures are closed and natural while social ones are open. However, a bridge can only be understood by considering how it reacts to traffic crossing it, and it must be sustained or repaired over time (Hitchins 2007). Boulding also separates humans from animals, which would not fit into more modern thinking.

Magee and de Weck (Magee and de Weck 2004) provide a comprehensive overview of sources on system classification such as (Maier and Rechtin 2009), (Paul 1998) and (Wasson 2006). They cover some methods for classifying natural systems, but their primary emphasis and value to the practice of systems engineer is in their classification method for human-designed, or man-made, systems. They examine many possible methods that include: degree of complexity, branch of the economy that produced the system, realm of existence (physical or in thought), boundary, origin, time dependence, system states, human involvement / system control, human wants, ownership and functional type. They conclude by proposing a functional classification method that sorts systems by their process (transform, transport, store, exchange, or control), and by the entity that they operate on matter, energy, information and value.

Systems of Systems

Systems can be grouped together to create more complex systems. In some cases it is sufficient to consider these systems as systems elements in a higher level system, as part of a system hierarchy.

However, there are cases where the groupings of system produce an entity that must be treated differently from an integrated system. The most common groupings of systems that have characteristics beyond a single integrated system are Systems of Systems (SoS) and Federations of Systems (FoS).

Maier examined the meaning of System of Systems in detail and used a characterization approach which emphasizes the independent nature of the system element, rather than “the commonly cited characteristics of systems-of-systems (complexity of the component systems and geographic distribution) which are not the appropriate taxonomic classifiers” (Maier 1998, 268).

Wherever independent systems are combined into groups the interaction between the systems adds a further complexity; specifically, by constraining how the resulting system can be changed or controlled. This dimension of complexity affects the management and control aspects of the systems approach.

A more detailed discussion of the different system grouping taxonomies developed by systems science can be found in Groupings of Systems.

Engineered Systems Classifications

The classification approaches discussed above have either been applied to all possible types of systems or have looked at how man-made systems differ from natural systems. The idea of an Engineered System (glossary) is to provide a focus on systems containing both technology and social or natural elements, developed for a defined purpose by an engineering life cycle. Engineered Systems:

- are created, used and sustained to achieve a purpose, goal or mission that is of interest to an enterprise, team, or an individual.
- require a commitment of resources for development and support.
- are driven by stakeholders (glossary) with multiple views on the use or creation of the system, or with some other stake in the system, its properties or existence.

- contain engineered hardware, software, people, services, or a combination of these.
- exist within an environment that impacts the characteristics, use, sustainment and creation of the system.

Engineered systems typically

- are defined by their purpose, goal or mission.
- have a life cycle (glossary) and evolution dynamics.
- may include human operators (interacting with the systems via processes) as well as other natural components that must be considered in the design and development of the system.
- are part of a system-of-interest hierarchy.

Historically,

> *Economists divide all economic activity into two broad categories, goods and services. Goods-producing industries are agriculture, mining, manufacturing, and construction; each of them creates some kind of tangible object. Service industries include everything else: banking, communications, wholesale and retail trade, all professional services such as engineering, computer software development, and medicine, nonprofit economic activity, all consumer services, and all government services, including defense and administration of justice....* (Encyclopedia Britannica 2011).

A product or service is developed and supported by an individual, team, or enterprise. For example, express package delivery is a service offered worldwide by many enterprises, public and private, small and large. These services might use vehicles, communications or software products, or a combination of the three as needed.

The nature of engineered systems has changed dramatically over the past several decades from systems dominated by hardware (mechanical and electrical) to systems dominated by software. In addition, systems that provide services, without delivering hardware or software, have become common as the need to obtain and use information has become greater. Recently, organizations have become sufficiently complex that the techniques that were demonstrated to work on hardware and software have been applied to the engineering of enterprises.

Three specific types of engineered system context are generally recognized in systems engineering: product system, service system and enterprise system.

Products and Product Systems

The word product (glossary) is defined as "a thing produced by labor or effort; or anything produced" (Oxford English Dictionary). In a commercial sense a product is anything which is acquired, owned and used by an enterprise (hardware, software, information, personnel, an agreement or contract to provide something, etc.).

Product systems are systems in which products are developed and delivered to the Acquirer (glossary) for the use of internal or external user. For product systems, the ability to provide the necessary capability (glossary) must be defined in the specifications for the hardware and software or the integrated system that will be provided to the acquiring enterprise.

Services and Service Systems

A service (glossary) can be simply defined as an act of help or assistance, or as any outcome required by one or more users which can be defined in terms of outcomes and quality of service without detail to how it is provided (e.g., transport, communications, protection, data processing, etc.) Services are processes, performances, or experiences that one person or organization does for the benefit of another, such as custom tailoring a suit; cooking a dinner to order; driving a limousine; mounting a legal defense; setting a broken bone; teaching a class; or running a business's information technology infrastructure and applications. In all cases, service involves deployment of knowledge and skills (competencies) that one person or organization has for the benefit of another (Lusch and Vargo 2006), often done as a single, customized job. In all cases, service requires substantial input from the customer or client (Sampson 2001). For example, how can a steak be customized unless the customer tells the waiter how the customer wants the

steak prepared?

A service system (glossary) is one that provides outcomes for a user without necessarily delivering hardware or software products to the service supplier. The hardware and software systems may be owned by a third party who is not responsible for the service. The use of service systems reduces or eliminates the need for acquirers to obtain capital equipment and software in order to obtain the capabilities needed to satisfy users.

Services have been part of the language of systems engineering (SE) for many years. The use of the term service system in more recent times is often associated with information systems, i.e.,

> *...unique features that characterize services – namely, services, especially emerging services, are information-driven, customer-centric, e-oriented, and productivity-focused.* (Tien and Berg 2003, 13)

A more detailed discussion of the system theory associated with service systems can be found in History of Systems Science.

Enterprises and Enterprise Systems

An enterprise (glossary) is one or more organizations or individuals sharing a definite mission, goals, and objectives to offer an output such as a product or service.

An enterprise system (glossary) consists of a purposeful combination (network) of interdependent resources (e.g., people; processes; organizations; supporting technologies; and funding) that interact with 1.) each other (e.g., to coordinate functions; share information; allocate funding; create workflows; and make decisions), and 2) their environment(s), to achieve business and operational goals through a complex web of interactions distributed across geography and time (Rebovich and White 2011, 4, 10, 34-35).

Both product and service systems require an enterprise system to create them and an enterprise to use the product system to deliver services either internally to the enterprise or externally to a broader community.

According to Maier's definition, an enterprise would not necessarily be called a system of systems (SoS) since the systems within the enterprise do not usually meet the criteria of operational and managerial independence. In fact, the whole purpose of an enterprise is to explicitly establish operational dependence between systems that the enterprise owns and/or operates in order to maximize the efficiency and effectiveness of the enterprise as a whole. Therefore, it is more proper to treat an enterprise system and an SoS as different types of things, with different properties and characteristics (DeRosa 2005).

Enterprise systems are unique, compared to product and service systems, in that they are constantly evolving; they rarely have detailed configuration controlled requirements; they typically have the goal of providing shareholder value and customer satisfaction, which are constantly changing and are difficult to verify; and they exist in a context (or environment) that is ill-defined and constantly changing.

The notion of enterprises and enterprise systems permeates Part 5 Enabling Systems Engineering.

System of Systems Capability

As discussed above, "system of systems" is a classification used for any system which contains elements which in some way can be considered as independent (Maier 1998). Any of the other three engineered system contexts described above may have some aspects of SoS to be considered across their life cycle. Similarly, capability is a concept relevant to all system contexts, relating to the real world outcomes which system users can achieve when the system is fully deployed in its operational environment.

The term **System of Systems Capability** is used here to describe an engineering context in which a number of enterprise, service and product systems are brought together dynamically to provide a capability which is beyond the scope of any individual enterprise.

Understanding the need for system of systems capability is a way of setting a broader problem context for the engineering of other systems. Both product and service systems may be engineered to both satisfy immediate

stakeholder needs and to have the potential to be used for the composition of SoS capabilities. Engineering at the Enterprise level can include an Enterprise Capability Management activity, in which possible SoS problems are explored and used to identify gaps in the enterprise's current product and service portfolio. (See the SEBoK, Part 4 Applications of Systems Engineering)

References

Works Cited

Aslaksen, E.W. 1996. *The Changing Nature of Engineering*. New York, NY, USA: McGraw-Hill.

Bertalanffy, L. von. 1968. *General System Theory*. New York, NY, USA: Brazillier.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Boulding, K. 1956 "General Systems Theory: Management Science, 2, 3 (Apr. 1956) pp.197-208; reprinted in General Systems, Yearbook of the Society for General Systems Research, vol. 1, 1956.

Checkland, P.B. 1999. *Systems Thinking, Systems Practice*. Chichester, UK: John Wiley & Sons Ltd.

Dictionary.com, s.v. "Taxonomy," Accessed December 3 2014 at Dictionary.com http://dictionary.reference.com/browse/taxonomy.

Encyclopedia Britannica, s.v. "Service Industry," Accessed December 3 2014 at Dictionary.com http://www.britannica.com/EBchecked/topic/535980/service-industry.

DeRosa, J. K. 2005. "Enterprise Systems Engineering." Air Force Association, Industry Day, Day 1, 4 August 2005, Danvers, MA.

Giachetti, R.E. 2009. *Design of Enterprise Systems: Theory, Architectures, and Methods.* Boca Raton, FL, USA: CRC Press.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* Hoboken, NJ, USA: Wiley.

Lusch, R.F. and S. L. Vargo (Eds). 2006. *The service-dominant logic of marketing: Dialog, debate, and directions*. Armonk, NY: ME Sharpe Inc.

Magee, C.L. and O.L. de Weck. 2004. "Complex System Classification". Proceedings of the 14th Annual International Symposium of the International Council on Systems Engineering, 20-24 June, 2004, Toulouse, France.

Maier, M. W. 1998. "Architecting Principles for Systems-of-Systems". *Systems Engineering*, 1(4): 267-84.

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting*, 3rd Ed.. Boca Raton, FL, USA: CRC Press.

Miller J. G. 1986. "Can Systems Theory Generate Testable Hypothesis?: From Talcott Parsons to Living Systems Theory?" *Systems Research.* 3:73-84.

Paul, A.S. 1998. "Classifying Systems." Proceedings of The 8th Annual International Council on Systems Engineering International Symposium, 26-30 July, 1998, Vancouver, BC, Canada.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press.

Sampson, S.E. 2001. *Understanding Service Businesses*. New York, NY, USA: John Wiley.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering*. 12(1): 13-38.

Wasson, C.S. 2006. *System Analysis, Design and Development.* Hoboken, NJ, USA: John Wiley and Sons.

Primary References

Checkland, P. B. 1999. *Systems Thinking, Systems Practice.* Chichester, UK: John Wiley & Sons.

Magee, C. L., O.L. de Weck. 2004. "Complex System Classification." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June 2004, Toulouse, France.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Groupings of Systems

This article forms part of the Systems Fundamentals knowledge area (KA). It expands on groups of system classification introduced in Types of Systems.

Systems can be grouped together to create more complex systems. In some cases, considering systems as system elements in a system hierarchy is sufficient. However, there are cases where the groupings of system produce an entity that must be treated differently from a single integrated system. This article explores the different descriptions of how and why system groupings might be considered.

System of Systems

The term “ system of systems” (SoS) is commonly used, but there is no widespread agreement on its exact meaning, or on how it can be distinguished from a conventional system. An extensive history of SoS is provided in “System-of-Systems Engineering Management: A Review of Modern History and a Path Forward” (Gorod & Boardman 2008). This paper provides a historical perspective for systems engineering from Brill (Brill 1998). The authors then provide a chronological history for SoS engineering from 1990 to 2008. Their history provides an extensive set of references to all of the significant papers and textbooks on SoS. Gorod and Boardman cite Maier as one of the most influential contributors to the study of SoS.

Maier examined the meaning of SoS in detail and used a characterization approach to create a definition (Maier 1998, 267-284). His definition has been adopted by many working in the field (AFSAB 2005). Maier provides this definition:

> *A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possess two additional properties:*

1. *Operational Independence of the Components: If the system-of-systems is disassembled into its component systems, the component systems must be able to usefully operate independently. That is, the components fulfill customer-operator purposes on their own.*
2. *Managerial Independence of the Components: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems.* (Maier 1998, 271)

Maier goes on further, saying that “the commonly cited characteristics of systems-of-systems (complexity of the component systems and geographic distribution) are not the appropriate taxonomic classifiers” (Maier 1998, 268). According to the Defense Acquisition Guide: "A SoS is defined as a set or arrangement of systems that results from independent systems integrated into a larger system that delivers unique capabilities" (DAU 2010, 4.1.4. System of Systems (SoS) Engineering). For further details on SoS, see the *Systems Engineering Guide for SoS* developed by the US Department of Defense (DoD) (DUS(AT) 2008).

Four kinds of SoS have been defined (Maier 1998; Dahmann and Baldwin 2008; DUS(AT) 2008; Dahmann, Lane, and Rebovich 2008):

1. ***Virtual****. Virtual SoS lack a central management authority and a centrally agreed upon purpose for the system-of-systems. Large-scale behavior emerges—and may be desirable—but this type of SoS must rely upon relatively invisible mechanisms to maintain it.*
2. ***Collaborative****. In collaborative SoS the component systems interact more or less voluntarily to fulfill agreed upon central purposes. The Internet is a collaborative system. The Internet Engineering Task Force works out standards but has no power to enforce them. The central players collectively decide how to provide or deny service, thereby providing some means of enforcing and maintaining standards.*
3. ***Acknowledged****. Acknowledged SoS have recognized objectives, a designated manager, and resources for the SoS; however, the constituent systems retain their independent ownership, objectives, funding, and development and sustainment approaches. Changes in the systems are based on collaboration between the SoS and the system.*
4. ***Directed****. Directed SoS are those in which the integrated system-of-systems is built and managed to fulfill specific purposes. It is centrally managed during long-term operation to continue to fulfill those purposes, as well as any new ones the system owners might wish to address. The component systems maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose* (DUS(AT) 2008, 4-5; and, Dahmann, Lane, and Rebovich 2008, 4; in reference to (Maier 1998; Dahmann and Baldwin 2008).)

The terms emergence (glossary) and emergent behavior are increasingly being used in SoS contexts, fueled, in part, by the movement to apply systems science and complexity theory to problems of large-scale, heterogeneous information technology based systems. In this context, a working definition of emergent behavior of a system is behavior which is unexpected or cannot be predicted by knowledge of the system’s constituent parts.

One of the leading authors in the area of SoS is Mo Jamshidi, who is the editor of a leading textbook (Jamshidi 2009) and articles such as “System of Systems Engineering – New Challenges for the 21st Century” (Jamshidi 2008). This article provides numerous references to papers that have examined the definition of SoS. The author selects six of the many potential definitions. His lead definition is

Systems of systems exist when there is a presence of a majority of the following five characteristics: operational and managerial independence; geographic distribution; emergent behavior; and evolutionary development. (Jamshidi 2008, 5; adapted from Sage and Cuppan 2001, 326)

Federation of Systems

Different from the SoS concept, but related to it in several ways, is the concept called federation of systems (FoS). This concept might apply when there is a very limited amount of centralized control and authority (Sage and Cuppan 2001). Each system in an FoS is very strongly in control of its own destiny, but “chooses” to participate in the FoS for its own good and the good of the “country,” so to speak. It is a coalition of the willing.

An FoS is generally characterized by significant autonomy, heterogeneity, and geographic distribution or dispersion (Krygiel 1999). Krygiel (1999) defined a taxonomy of systems showing the relationships among conventional systems, SoSs, and FoSs. This taxonomy has three dimensions: autonomy; heterogeneity; and dispersion. An FoS would have a larger value on each of these three dimensions than a non-federated SoS. An enterprise system, as described in Types of Systems, could be considered to be an FoS if it rates highly on these three dimensions. However, it is possible for an enterprise to have components that are not highly autonomous, that are relatively homogenous, and are geographically close together. Therefore, it would be a mistake to say that an enterprise is necessarily the same as an FoS.

Handy (1992) describes a federalist approach called “New Federalism”, which identifies the need for structuring of loosely coupled organizations to help them adapt to the rapid changes inherent in the Information Age. This leads to the need for virtual organizations where alliances can be quickly formed to handle the challenges of newly identified threats and a rapidly changing marketplace (Handy 1995). Handy sets out to define a number of federalist political principles that could be applicable to an FoS. Handy’s principles have been tailored to the domain of systems engineering and management by Sage and Cuppan (2001).

Families of Systems

The Defense Acquisition University (DAU 2010, 4.1.4. System of Systems (SoS) Engineering) defines families of systems as:

> *A grouping of systems having some common characteristic(s). For example, each system in a family of systems may belong to a domain or product line (e.g., a family of missiles, aircraft, or situation awareness systems). In general, a family of systems is not considered to be a system per se because it does not necessarily create capability beyond the additive sum of the individual capabilities of its member systems. A family of systems lacks the synergy of a SoS. The family of systems does not acquire qualitatively new properties as a result of the grouping. In fact, the member systems may not be connected into a whole.* (DAU 2010)

Very few papers have been written that address families of systems or compare them to systems of systems.

James Clark provides a view that a family of systems is equivalent to a product line:

> *By family, we mean a product-line or domain, wherein some assets are re-used un-modified; some assets are modified, used, and re-used later; and some assets are developed new, used, and re-used later. Product-lines are the result.* (Clark 2008)

References

Works Cited

AFSAB. 2005. *Report on Domain Integration.* Washington, D.C.: U.S. Air Force Scientific Advisory Board/U.S. Air Force. SAB-TR-05-03.

Brill, J. H. 1998. "Systems Engineering – A Retrospective View." *Systems Engineering*. 1(4): 258-266.

Clark, J. 2008. "System of Systems Engineering and Family of Systems Engineering From a Standards, V-Model, and Dual-V Model Perspective." Proceedings of the 18th Annual International Council on Systems Engineering International Symposium, 15-19 June, 2008, Utrecht, The Netherlands.

Dahmann, J., and K. Baldwin. 2008. "Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering." Paper presented at IEEE Systems Conference, 7-10 April, Montreal, Canada.

Dahmann, J.S., J.A. Lane, and G. Rebovich. 2008. "Systems Engineering for Capabilities." *CROSSTALK: The Journal of Defense Software Engineering*. (November 2008): 4-9.

DAU. February 19, 2010. *Defense acquisition guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense.

DUS(AT). 2008. *Systems Engineering Guide for Systems of Systems," version 1.0. Washington, DC, USA: Deputy Under Secretary of Defense for Acquisition and Technology (DUS(AT))/U.S. Department of Defense (DoD).*

Gorod, A., B. Sauser, and J. Boardman. 2008. "System-of-Systems Engineering Management: A Review of Modern History and a Path Forward". IEEE *Systems Journal*, 2(4): 484-499.

Handy, C. 1995. "How Do You Manage People Whom You Do Not See? Trust and the Virtual Organization." *Harvard Business Review.' 73(3) (May-June): 40-50.*

Handy, C. 1992. "Balancing Corporate Power: A New Federalist Paper". *Harvard Business Review.* 70(6) (November/December): 59-72.

Jain, P. and Dickerson, C. 2005. "Family-of-Systems Architecture Analysis Technologies." Proceedings of the 15th Annual International Council on Systems Engineering International Symposium, 10-15, July 2005, Rochester, NY, USA.

Jamshidi, M. "Theme of the IEEE SMC 2005" in IEEE SMC 2005 - International Conference on Systems, Man, and Cybernetics. Accessed 11 September 2011. Available at: http://ieeesmc2005.unm.edu.

Jamshidi, M. (ed.). 2009. *Systems of Systems Engineering – Innovations for the 21st Century*. Hoboken, NJ, USA: John Wiley and Sons.

Jamshidi, M. 2008. "System of Systems Engineering – New Challenges for the 21st Century". IEEE *Aerospace and Electronic Systems Magazine.* 23(5) (May 2008): 4-19.

Krygiel, A.J. 1999. *Behind the Wizard's Curtain: An Integration Environment for a System of Systems.* Arlington, VA, USA: C4ISR Cooperative Research Program (CCRP).

Maier, M.W. 1998. "Architecting Principles for Systems-of-Systems". *Systems Engineering*, 1(4): 267-84.

Sage, A., and C. Cuppan. 2001. "On the Systems Engineering and Management of Systems of Systems and Federations of Systems". *Information-Knowledge-Systems Management Journal.* 2(4) (December 2001): 325-45.

Primary References

Gorod, A., B. Sauser, and J. Boardman. 2008. "System-of-Systems Engineering Management: A Review of Modern History and a Path Forward." IEEE *Systems Journal*. 2(4): 484-499.

Jamshidi, M. (ed). 2009. *Systems of Systems Engineering – Innovations for the 21st Century.* Hoboken, NJ: Wiley and Sons.

Jamshidi, M. 2008. "System of Systems Engineering – New Challenges for the 21st Century." IEEE *Aerospace and Electronic Systems Magazine.* 23(5) (May 2008): 4-19.

Maier, M.W. 1998. "Architecting Principles for Systems-of-Systems". *Systems Engineering.* 1(4): 267-84.

Sage, A. and C. Cuppan. 2001. "On the Systems Engineering and Management of Systems of Systems and Federations of Systems." *Information-Knowledge-Systems Management Journal.* 2(4) (December 2001): 325-45.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 12 October 2018

Complexity

This article is part of the Systems Fundamentals knowledge area (KA). It gives the background and an indication of current thinking on complexity and how it influences systems engineering (SE) practice.

Complexity is one of the most important and difficult to define system concepts. Is a system's complexity in the eye of the beholder, or is there inherent complexity in how systems are organized? Is there a single definitive definition of complexity and, if so, how can it be assessed and measured? This topic will discuss how these ideas relate to the general definitions of a system given in What is a System?, and in particular to the different engineered system contexts. This article is closely related to the emergence topic that follows it.

Defining System Complexity

Complexity has been considered by a number of authors, from various perspectives, some of the discussions of complexity relevant to systems are described in the final section of this article. Sheard and Mostashari (Sheard and Mostashari 2011) synthesize many of these ideas to categorize complexity as follows:

1. **Structural Complexity** looks at the system elements and relationships. In particular, structural complexity looks at how many different ways system elements can be combined. Thus, it is related to the potential for the system to adapt to external needs.
2. **Dynamic Complexity** considers the complexity which can be observed when systems are used to perform particular tasks in an environment. There is a time element to dynamic complexity. The ways in which systems interact in the short term is directly related to system behavior; the longer term effects of using systems in an environment is related to system evolution.
3. **Socio-political Complexity** considers the effect of individuals or groups of people on complexity. People-related complexity has two aspects. One is related to the perception of a situation as complex or not, due to multiple stakeholder viewpoints within a system context and social or cultural biases which add to the wider influences on a system context. The other involves either the “irrational” behavior of an individual or the swarm behavior of many people behaving individually in ways that make sense; however, the emergent behavior is unpredicted and perhaps counterproductive. This latter type is based on the interactions of the people according to their various

interrelationships and is often graphed using systems dynamics formalisms.

Thus, complexity is a measure of how difficult it is to understand how a system will behave or to predict the consequences of changing it. It occurs when there is no simple relationship between what an individual element does and what the system as a whole will do, and when the system includes some element of adaptation or problem solving to achieve its goals in different situations. It can be affected by objective attributes of a system such as by the number, types of and diversity of system elements and relationships, or by the subjective perceptions of system observers due to their experience, knowledge, training, or other sociopolitical considerations.

This view of complex systems provides insight into the kind of system for which systems thinking and a systems approach is essential.

Complexity and Engineered Systems

The different perspectives on complexity are not independent when considered across a systems context. The Structural complexity of a system-of-interest (SoI) may be related to dynamic complexity when the SoI also functions as part of a wider system in different problem scenarios. People are involved in most system contexts, as part of the problem situation, as system elements and part of the operating environment. The human activity systems which we create to identify, design, build and support an engineered system and the wider social and business systems in which they sit are also likely to be complex and affect the complexity of the systems they produce and use.

Sheard and Mostashari (Sheard and Mostashari 2011) show the ways different views of complexity map onto product system, service system and enterprise system contexts, as well as to associated development and sustainment systems and project organizations. Ordered systems occur as system components and are the subject of traditional engineering. It is important to understand the behaviors of such systems when using them in a complex system. One might also need to consider both truly random or chaotic natural or social systems as part of the context of an engineered system. The main focus for systems approaches is **organized complexity** (see below). This kind of complexity cannot be dealt with by traditional analysis techniques, nor can it be totally removed by the way we design or use solutions. A systems approach must be able to recognise and deal with such complexity across the life of the systems it interacts with..

Sillitto (Sillitto 2014) considers the link between the types of system complexity and system architecture. The ability to understand, manage and respond to both objective and subjective complexity in the problem situation, the systems we develop or the systems we use to develop and sustain them is a key component of the Systems Approach Applied to Engineered Systems and hence to the practice of systems engineering.

Origins and Characteristics of Complexity

This section describes some of the prevailing ideas on complexity. Various authors have used different language to express these ideas. While a number of common threads can be seen, some of the ideas take different viewpoints and may be contradictory in nature.

One of the most widely used definitions of complexity is the degree of difficulty in predicting the properties of a system if the properties of the system's parts are given (generally attributed to Weaver). This, in turn, is related to the number of elements and connections between them. Weaver (Weaver 1948) relates complexity to types of elements and how they interact. He describes simplicity as problems with a finite number of variables and interaction, and identifies two kinds of complexity:

1. **Disorganized Complexity** is found in a system with many loosely coupled, disorganized and equal elements, which possesses certain average properties such as temperature or pressure. Such a system can be described by “19th Century” statistical analysis techniques.

2. **Organized Complexity** can be found in a system with many strongly coupled, organized and different elements which possess certain emergent properties and phenomena such as those exhibited by economic, political or social systems. Such a system can not be described well by traditional analysis techniques.

Weaver's ideas about this new kind of complex problem are one of the foundational ideas of systems thinking. (See also Systems Thinking.)

Later authors, such as Flood and Carson (Flood and Carson 1993) and Lawson (Lawson 2010), expand organized complexity to systems which have been organized into a structure intended to be understood and thus amenable to engineering and life cycle management (Braha et al. 2006). They also suggest that disorganized complexity could result from a heterogeneous complex system evolving without explicit architectural control during its life (complexity creep). This is a different use of the terms organized and disorganized to that used by Weaver. Care should be taken in mixing these ideas

Complexity should not be confused with "complicated". Many authors make a distinction between ordered and disordered collections of elements.

Ordered systems have fixed relationships between elements and are not adaptable. Page (Page 2009) cites a watch as an example of something which can be considered an ordered system. Such a system is complicated, with many elements working together. Its components are based on similar technologies, with clear mapping between form and function. If the operating environment changes beyond prescribed limits, or one key component is removed, the watch will cease to perform its function.

In common usage, chaos is a state of disorder or unpredictability characterized by elements which are not interconnected and behave randomly with no adaptation or control. Chaos Theory (Kellert 1993) is applied to certain dynamic systems (e.g., the weather) which, although they have structure and relationships, exhibit unpredictable behavior. These systems may include aspects of randomness but can be described using deterministic models from which their behavior can be described given a set of initial conditions. However, their structure is such that (un-measurably) small perturbations in inputs or environmental conditions may result in unpredictable changes in behavior. Such systems are referred to as deterministically chaotic or, simply, chaotic systems. Simulations of chaotic systems can be created and, with increases in computing power, reasonable predictions of behavior are possible at least some of the time.

On a spectrum of order to complete disorder complexity is somewhere in the middle, with more flexibility and change than complete order and more stability than complete disorder (Sheard and Mostashari 2009).

Complex systems may evolve “to the edge of chaos”, resulting in systems which can appear deterministic but which exhibit counter intuitive behavior compared to that of more ordered systems. The statistics of chance events in a complex system are often characterized by a power-law distribution, the “signature of complexity” (Sheard 2005). The power-law distribution is found in a very wide variety of natural and man-made phenomena, and it means that the probability of a low probability—large impact event is much higher than a Gaussian distribution would suggest. Such a system may react in a non-linear way to exhibit abrupt phase changes. These phase changes can be either reversible or irreversible. This has a major impact on engineered systems in terms of the occurrence, impact and public acceptance of risk and failure.

Objective complexity is an attribute of complex systems and is a measure of where a system sits on this spectrum. It is defined as the extent to which future states of the system cannot be predicted with certainty and precision, regardless of our knowledge of current state and history. Subjective complexity is a measure of how easy it is for an observer to understand a system or predict what it will do next. As such, it is a function of the perspective and comprehension of each individual. It is important to be prepared to mitigate subjective complexity with consistent, clear communication and strong stakeholder engagement (Sillitto 2009).

The literature has evolved to a fairly consistent definition of the characteristics of system elements and relationships for objective systems complexity. The following summary is given by Page (Page 2009):

1. **Independence**: Autonomous system elements which are able to make their own decisions; influenced by information from other elements and the adaptability algorithms it carries with it (Sheard and Mostashari 2009).
2. **Interconnectedness**: System elements connect via a physical connection, shared data or simply a visual awareness of where the other elements are and what they are doing, as in the case of the flock of geese or the squadron of aircraft.
3. **Diversity**: System elements which are either technologically or functionally different in some way. For example, elements may be carrying different adaptability algorithms.
4. **Adaptability**: Self-organizing system element which can do what it wants to do to support itself or the entire system in response to their environment (Sheard and Mostashari 2009). Adaptability is often achieved by human elements but can be achieved with software. Pollock and Hodgson (2004) describe how this can be done in a variety of complex system types, including power grids and enterprise systems.

Due to the variability of human behavior as part of a system and the perceptions of people outside the system, the inclusion of people in a system is often a factor in their complexity. People may be viewed as observing systems or as system elements which contribute to the other types of complexity (Axelrod and Cohen 1999). The rational or irrational behavior of individuals in particular situations is a vital factor in respect to complexity (Kline 1995). Some of this complexity can be reduced through education, training and familiarity with a system. Some is irreducible, and must be managed as part of a problem or solution. Checkland (Checkland 1999) argues that a group of stakeholders will have its own world views which lead them to form different, but equally valid, understandings of a system context. These differences cannot be explained away or analyzed out, and must be understood and considered in the formulation of problems and the creation of potential solutions.

Warfield (Warfield 2006) developed a powerful methodology for addressing complex issues, particularly in the socio-economic field, based on a relevant group of people developing an understanding of the issue in the form of a set of interacting problems - what he called the “problematique”. The complexity is then characterized via several measures, such as the number of significant problems, their interactions and the degree of consensus about the nature of the problems. What becomes clear is that how, why, where and by whom a system is used may all contribute to its perceived complexity.

Sheard and Mostashari (Sheard and Mostashari 2011) sort the attributes of complexity into causes and effects. Attributes that cause complexity include being non-linear; emergent; chaotic; adaptive; tightly coupled; self-organized; decentralized; open; political (as opposed to scientific); and multi-scale; as well as having many pieces. The effects of those attributes which make a system be perceived as complex include being uncertain; difficult to understand; unpredictable; uncontrollable; unstable; unrepairable; unmaintainable and costly; having unclear cause and effect; and taking too long to build.

References

Works Cited

Axelrod, R. and M. Cohen. 1999. *Harnessing Complexity: Organizational Implications of a Scientific Frontier*. New York, NY, USA: Simon and Schuster.

Braha, D., A. Minai, and Y. Bar-Yam (eds.). 2006. *Complex Engineered Systems: Science Meets Technology*. New York, NY, USA: Springer.

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Flood, R. L., and E.R. Carson. 1993. *Dealing with Complexity: An Introduction to The Theory and Application of Systems Science", 2nd ed*. New York, NY, USA: Plenum Press.

Lawson, H. W. 2010. *A Journey Through the Systems Landscape*. Kings College, UK: College Publications.

Kellert, S. 1993. *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*, Chicago, IL, USA: University of Chicago Press. p. 32.

Kline, S. 1995. *Foundations of Multidisciplinary Thinking*. Stanford, CA, USA: Stanford University Press.

Page, Scott E. 2009. *Understanding Complexity*. Chantilly, VA, USA: The Teaching Company.

Pollock, J.T. and R. Hodgson. 2004. *Adaptive Information*. Hoboken, NJ, USA: John Wiley & Sons.

Senge, P.M. 1990. *The Fifth Discipline: The Art & Practice of The Learning Organization*. New York, NY, USA: Doubleday/Currency.

Sheard, S.A. 2005. "Practical Applications of Complexity Theory for Systems Engineers". *Proceedings of the Fifteenth Annual International Council on Systems Engineering*. Volume 15 Issue 1.

Sheard, S.A. and A. Mostashari. 2009. "Principles of Complex Systems for Systems Engineering." *Systems Engineering*, 12(4): 295-311.

Sheard, SA. and A. Mostashari. 2011. "Complexity Types: From Science to Systems Engineering." Proceedings of the 21st Annual of the International Council on Systems Engineering (INCOSE) International Symposium, 20-23 June 2011, Denver, Colorado, USA.

Sillitto H 2014, "Architecting Systems - Concepts, Principles and Practice", College Publications 2014

Warfield, J.N. 2006. *An Introduction to Systems Science*. London, UK: World Scientific Publishing.

Weaver, W. 1948. "Science and Complexity." *American Science.* 36: 536-544.

Primary References

Flood, R. L., & E.R. Carson. 1993. *Dealing with Complexity: An Introduction to The Theory and Application of Systems Science*, 2nd ed. New York, NY, USA: Plenum Press.

Page, Scott E. 2009. *Understanding Complexity*. Chantilly, VA, USA: The Teaching Company.

Sheard, S.A. and A. Mostashari. 2009. "Principles of Complex Systems for Systems Engineering". *Systems Engineering*, 12(4): 295-311.

Additional References

Ashby, W.R. 1956. *An Introduction to Cybernetics*. London, UK: Chapman and Hall.

Aslaksen, E.W. 2004. "System Thermodynamics: A Model Illustrating Complexity Emerging from Simplicity". *Systems Engineering*, 7(3). Hoboken, NJ, USA: Wiley.

Aslaksen, E.W. 2009. *Engineering Complex Systems: Foundations of Design in the Functional Domain*. Boca Raton, FL, USA: CRC Press.

Aslaksen, E.W. 2011. "Elements of a Systems Engineering Ontology". Proceedings of SETE 2011, Canberra, Australia.

Eisner, H. 2005. *Managing Complex Systems: Thinking Outside the Box*. Hoboken, NJ, USA: John Wiley & Sons.

Jackson, S., D. Hitchins, and H. Eisner. 2010. What is the Systems Approach? INCOSE *Insight* 13(1) (April 2010): 41-43.

MITRE. 2011. "Systems Engineering Strategies for Uncertainty and Complexity." *Systems Engineering Guide.* Accessed 9 March 2011 at http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ comprehensive_viewpoint/sys_engineering_strategies_uncertainty_complexity.html.

Ryan, A. 2007. "Emergence Is Coupled to Scope, Not Level, Complexity". A condensed version appeared in INCOSE *Insight*, 11(1) (January 2008): 23-24.

Sillitto H.G. 2009. "On Systems Architects and Systems Architecting: Some Thoughts on Explaining The Art and Science of System Architecting." Proceedings of the 19th Annual International Council on Systems Engineering

(INCOSE) International Symposium, 20-23 July 2009, Singapore.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 12 October 2018

Emergence

This topic forms part of the Systems Fundamentals knowledge area (KA). It gives the background to some of the ways in which emergence has been described, as well as an indication of current thinking on what it is and how it influences systems engineering (SE) practice. It will discuss how these ideas relate to the general definitions of systems given in What is a System?; in particular, how they relate to different engineered system contexts. This topic is closely related to the complexity topic that precedes it.

Emergence is a consequence of the fundamental system concepts of holism and interaction (Hitchins 2007, 27). System wholes have behaviors and properties arising from the organization of their elements and their relationships, which only become apparent when the system is placed in different environments.

Questions that arise from this definition include: What kinds of systems exhibit different kinds of emergence and under what conditions? Can emergence be predicted, and is it beneficial or detrimental to a system? How do we deal with emergence in the development and use of engineered systems? Can it be planned for? How?

There are many varied and occasionally conflicting views on emergence. This topic presents the prevailing views and provides references for others.

Overview of Emergence

As defined by Checkland, emergence is “the principle that entities exhibit properties which are meaningful only when attributed to the whole, not to its parts.” (Checkland 1999, 314). Emergent system behavior can be viewed as a consequence of the interactions and relationships between system elements rather than the behavior of individual elements. It emerges from a combination of the behavior and properties of the system elements and the systems structure or allowable interactions between the elements, and may be triggered or influenced by a stimulus from the systems environment.

Emergence is common in nature. The pungent gas ammonia results from the chemical combination of two odorless gases, hydrogen and nitrogen. As individual parts, feathers, beaks, wings, and gullets do not have the ability to overcome gravity. Properly connected in a bird, however, they create the emergent behavior of flight. What we refer to as “self-awareness” results from the combined effect of the interconnected and interacting neurons that make up the brain (Hitchins 2007, 7).

Hitchins also notes that technological systems exhibit emergence. We can observe a number of levels of outcome which arise from interaction between elements in an engineered system context. At a simple level, some system outcomes or attributes have a fairly simple and well defined mapping to their elements; for example, center of gravity or top speed of a vehicle result from a combination of element properties and how they are combined. Other behaviors can be associated with these simple outcomes, but their value emerges in complex and less predictable ways across a system. The single lap performance of a vehicle around a track is related to center of gravity and speed; however, it is also affected by driver skill, external conditions, component ware, etc. Getting the 'best' performance from a vehicle can only be achieved by a combination of good design and feedback from real laps under race conditions.

There are also outcomes which are less tangible and which come as a surprise to both system developers and users. How does lap time translate into a winning motor racing team? Why is a sports car more desirable to many than other vehicles with performances that are as good or better?

Emergence can always be observed at the highest level of system. However, Hitchins (2007, 7) also points out that to the extent that the systems elements themselves can be considered as systems, they also exhibit emergence. Page (Page 2009) refers to emergence as a “macro-level property.” Ryan (Ryan 2007) contends that emergence is coupled to scope rather than system hierarchical levels. In Ryan’s terms, scope has to do with spatial dimensions (how system elements are related to each other) rather than hierarchical levels.

Abbott (Abbott 2006) does not disagree with the general definition of emergence as discussed above. However, he takes issue with the notion that emergence operates outside the bounds of classical physics. He says that “such higher-level entities…can always be reduced to primitive physical forces.”

Bedau and Humphreys (2008) and Francois (2004) provide comprehensive descriptions of the philosophical and scientific background of emergence.

Types of Emergence

A variety of definitions of types of emergence exists. See Emmeche et al. (Emmeche et al. 1997), Chroust (Chroust 2003) and O’Connor and Wong (O’Connor and Wong 2006) for specific details of some of the variants. Page (Page 2009) describes three types of emergence: "simple", "weak", and "strong".

According to Page, **simple emergence** is generated by the combination of element properties and relationships and occurs in non-complex or “ordered” systems (see Complexity) (2009). To achieve the emergent property of “controlled flight” we cannot consider only the wings, or the control system, or the propulsion system. All three must be considered, as well as the way these three are interconnected-with each other, as well as with all the other parts of the aircraft. Page suggests that simple emergence is the only type of emergence that can be predicted. This view of emergence is also referred to as synergy (Hitchins 2009).

Page describes **weak emergence** as expected emergence which is desired (or at least allowed for) in the system structure (2009). However, since weak emergence is a product of a complex system, the actual level of emergence cannot be predicted just from knowledge of the characteristics of the individual system components.

The term **strong emergence** is used to describe unexpected emergence; that is, emergence not observed until the system is simulated or tested or, more alarmingly, until the system encounters in operation a situation that was not anticipated during design and development.

Strong emergence may be evident in failures or shutdowns. For example, the US-Canada Blackout of 2003 as described by the US-Canada Power System Outage Task Force (US-Canada Power Task Force 2004) was a case of cascading shutdown that resulted from the design of the system. Even though there was no equipment failure, the shutdown was systemic. As Hitchins points out, this example shows that emergent properties are not always beneficial (Hitchins 2007, 15).

Other authors make a different distinction between the ideas of strong, or unexpected, emergence and unpredictable emergence:

- Firstly, there are the unexpected properties that could have been predicted but were not considered in a systems development: "Properties which are unexpected by the observer because of his incomplete data set, with regard to the phenomenon at hand" (Francois, C. 2004, 737). According to Jackson et al. (Jackson et al. 2010), a desired level of emergence is usually achieved by iteration. This may occur as a result of evolutionary processes, in which element properties and combinations are "selected for", depending on how well they contribute to a systems effectiveness against environmental pressures or by iteration of design parameters through simulation or build/test cycles. Taking this view, the specific values of weak emergence can be refined and examples of strong emergence can be considered in subsequent iterations so long as they are amenable to analysis.
- Secondly, there are unexpected properties which cannot be predicted from the properties of the system’s components: "Properties which are, in and of themselves, not derivable a priori from the behavior of the parts of the system" (Francois, C. 2004, 737). This view of emergence is a familiar one in social or natural sciences, but

more controversial in engineering. We should distinguish between a theoretical and a practical unpredictability (Chroust 2002). The weather forecast is theoretically predictable, but beyond certain limited accuracy practically impossible due to its chaotic nature. The emergence of consciousness in human beings cannot be deduced from the physiological properties of the brain. For many, this genuinely unpredictable type of complexity has limited value for engineering. (See **Practical Considerations** below.)

A type of system particularly subject to strong emergence is the System of Systems (SoS) (glossary). The reason for this is that the SoS, by definition, is composed of different systems that were designed to operate independently. When these systems are operated together, the interaction among the parts of the system is likely to result in unexpected emergence. Chaotic or truly unpredictable emergence is likely for this class of systems.

Emergent Properties

Emergent properties can be defined as follows: “A property of a complex system is said to be ‘emergent’ [in the case when], although it arises out of the properties and relations characterizing its simpler constituents, it is neither predictable from, nor reducible to, these lower-level characteristics” (Honderich 1995, 224).

All systems can have emergent properties which may or may not be predictable or amenable to modeling, as discussed above. Much of the literature on complexity includes emergence as a defining characteristic of complex systems. For example, Boccara (Boccara 2004) states that “The appearance of emergent properties is the single most distinguishing feature of complex systems”. In general, the more ordered a systems is, the easier its emergent properties are to predict. The more complex a system is, the more difficult predicting its emergent properties becomes.

Some practitioners use the term “emergence” only when referring to “strong emergence”. These practitioners refer to the other two forms of emergent behavior as synergy or “system level behavior” (Chroust 2002). Taking this view, we would reserve the term "Emergent Property" for unexpected properties, which can be modeled or refined through iterations of the systems development.

Unforeseen emergence causes nasty shocks. Many believe that the main job of the systems approach is to prevent undesired emergence in order to minimize the risk of unexpected and potentially undesirable outcomes. This review of emergent properties is often specifically associated with identifying and avoiding system failures (Hitchins 2007).

Good SE isn't just focused on avoiding system failure, however. It also involves maximizing opportunity by understanding and exploiting emergence in engineered systems to create the required system level characteristics from synergistic interactions between the components, not just from the components themselves (Sillitto 2010).

One important group of emergent properties include properties such as agility and resilience. These are critical system properties that are not meaningful except at the whole system level.

Practical Considerations

As mentioned above, one way to manage emergent properties is through iteration. The requirements to iterate the design of an engineered system to achieve desired emergence results in a design process are more lengthy than those needed to design an ordered system. Creating an engineered system capable of such iteration may also require a more configurable or modular solution. The result is that complex systems may be more costly and time-consuming to develop than ordered ones, and the cost and time to develop is inherently less predictable.

Sillitto (2010) observes that “engineering design domains that exploit emergence have good mathematical models of the domain, and rigorously control variability of components and subsystems, and of process, in both design and operation”. The iterations discussed above can be accelerated by using simulation and modeling, so that not all the iterations need to involve building real systems and operating them in the real environment.

The idea of domain models is explored further by Hybertson in the context of general models or patterns learned over time and captured in a model space (Hybertson 2009). Hybertson states that knowing what emergence will

appear from a given design, including side effects, requires hindsight. For a new type of problem that has not been solved, or a new type of system that has not been built, it is virtually impossible to predict emergent behavior of the solution or system. Some hindsight, or at least some insight, can be obtained by modeling and iterating a specific system design; however, iterating the design within the development of one system yields only limited hindsight and often does not give a full sense of emergence and side effects.

True hindsight and understanding comes from building multiple systems of the same type and deploying them, then observing their emergent behavior in operation and the side effects of placing them in their environments. If those observations are done systematically, and the emergence and side effects are distilled and captured in relation to the design of the systems — including the variations in those designs — and made available to the community, then we are in a position to predict and exploit the emergence.

Two factors are discovered in this type of testing environment: what works (that is, what emergent behavior and side effects are desirable); and what does not work (that is, what emergent behavior and side effects are undesirable). What works affirms the design. What does not work calls for corrections in the design. This is why multiple systems, especially complex systems, must be built and deployed over time and in different environments; to learn and understand the relations among the design, emergent behavior, side effects, and environment.

These two types of captured learning correspond respectively to patterns and “antipatterns”, or patterns of failure, both of which are discussed in a broader context in the Principles of Systems Thinking and Patterns of Systems Thinking topics.

The use of iterations to refine the values of emergent properties, either across the life of a single system or through the development of patterns encapsulating knowledge gained from multiple developments, applies most easily to the discussion of strong emergence above. In this sense, those properties which can be observed but cannot be related to design choices are not relevant to a systems approach. However, they can have value when dealing with a combination of engineering and managed problems which occur for system of systems contexts (Sillitto 2010). (See Systems Approach Applied to Engineered Systems.)

References

Works Cited

Abbott, R. 2006. "Emergence Explained: Getting Epiphenomena to Do Real Work". *Complexity.* 12(1) (September-October): 13-26.

Bedau, M.A. and P. Humphreys, P. (eds.). 2008. "Emergence" In Contemporary Readings in Philosophy and Science. Cambridge, MA, USA: The MIT Press.

Boccara, N. 2004. *Modeling Complex Systems.* New York: Springer-Verlag.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Chroust. G. 2002. "Emergent Properties in Software Systems." 10th Interdisciplinary Information Management Talks; Hofer, C. and Chroust, G. (eds.). Verlag Trauner Linz, pages 277-289.

Chroust, G., C. Hofer, C. Hoyer (eds.). 2005. *The Concept of Emergence in Systems Engineering." The 12th Fuschl Conversation, April 18-23, 2004, Institute for Systems Engineering and Automation, Johannes Kepler University Linz. pp. 49-60.*

Emmeche, C., S. Koppe, and F. Stjernfelt. 1997. "Explaining Emergence: Towards an Ontology of Levels." *Journal for General Philosophy of Science.* 28: 83-119 (1997). Accessed December 3 2014 at Claus Emmeche http://www.nbi.dk/~emmeche/coPubl/97e.EKS/emerg.html.

Francois, C. 2004. *International Encyclopedia of Systems and Cybernetics*, 2nd edition, 2 volumes. K.G.Saur, Munchen.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology*. Hoboken, NJ, USA: John Wiley & Sons.

Honderich. T. 1995. *The Oxford Companion to Philosophy*. New York: Oxford University Press.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Auerbach/CRC Press, Boca Raton, FL.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1) (April 2010): 41-43.

O'Connor, T. and H. Wong. 2006. "Emergent Properties". *Stanford Encyclopedia of Philosophy*. Accessed December 3 2014 at Stanford Encyclopedia of Philosophy http://plato.stanford.edu/entries/properties-emergent/.

Page, S.E. 2009. *Understanding Complexity.* The Great Courses. Chantilly, VA, USA: The Teaching Company.

Ryan, A. 2007. "Emergence is Coupled to Scope, Not Level." *Complexity.* 13(2) (November-December).

Sillitto, H.G. 2010. "Design Principles for Ultra-Large-Scale Systems". Proceedings of the 20th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 2010, Chicago, IL, USA, reprinted in "The Singapore Engineer", April 2011.

US-Canada Power System Outage Task Force. 2004. *Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*. April, 2004. Washington-Ottawa. Accessed December 3 2014 at US Department of Energy http:/ / energy. gov/ oe/ downloads/ blackout-2003-final-report-august-14-2003-blackout-united-states-and-canada-causes-and

Primary References

Emmeche, C., S. Koppe, and F. Stjernfelt. 1997. "Explaining Emergence: Towards an Ontology of Levels." *Journal for General Philosophy of Science*, 28: 83-119 (1997). http://www.nbi.dk/~emmeche/coPubl/97e.EKS/emerg.html.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology*. Hoboken, NJ, USA: John Wiley & Sons.

Page, S. E. 2009. *Understanding Complexity*. The Great Courses. Chantilly, VA, USA: The Teaching Company.

Additional References

Sheard, S.A. and A. Mostashari. 2008. "Principles of Complex Systems for Systems Engineering." *Systems Engineering*. 12: 295-311.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Systems Science

Systems Science

This knowledge area (KA) provides a guide to some of the major developments in systems science which is an interdisciplinary field of science that studies the nature of complex systems in nature, society, and engineering.

This is part of the wider systems knowledge which can help to provide a common language and intellectual foundation; and make practical systems concepts, principles, patterns and tools accessible to systems engineering (SE) as discussed in the Introduction to Part 2.

Topics

Each part of the SEBoK is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- History of Systems Science
- Systems Approaches

Introduction

The following diagram summarizes the relationships between Systems Science (glossary) and other sections of the SEBoK

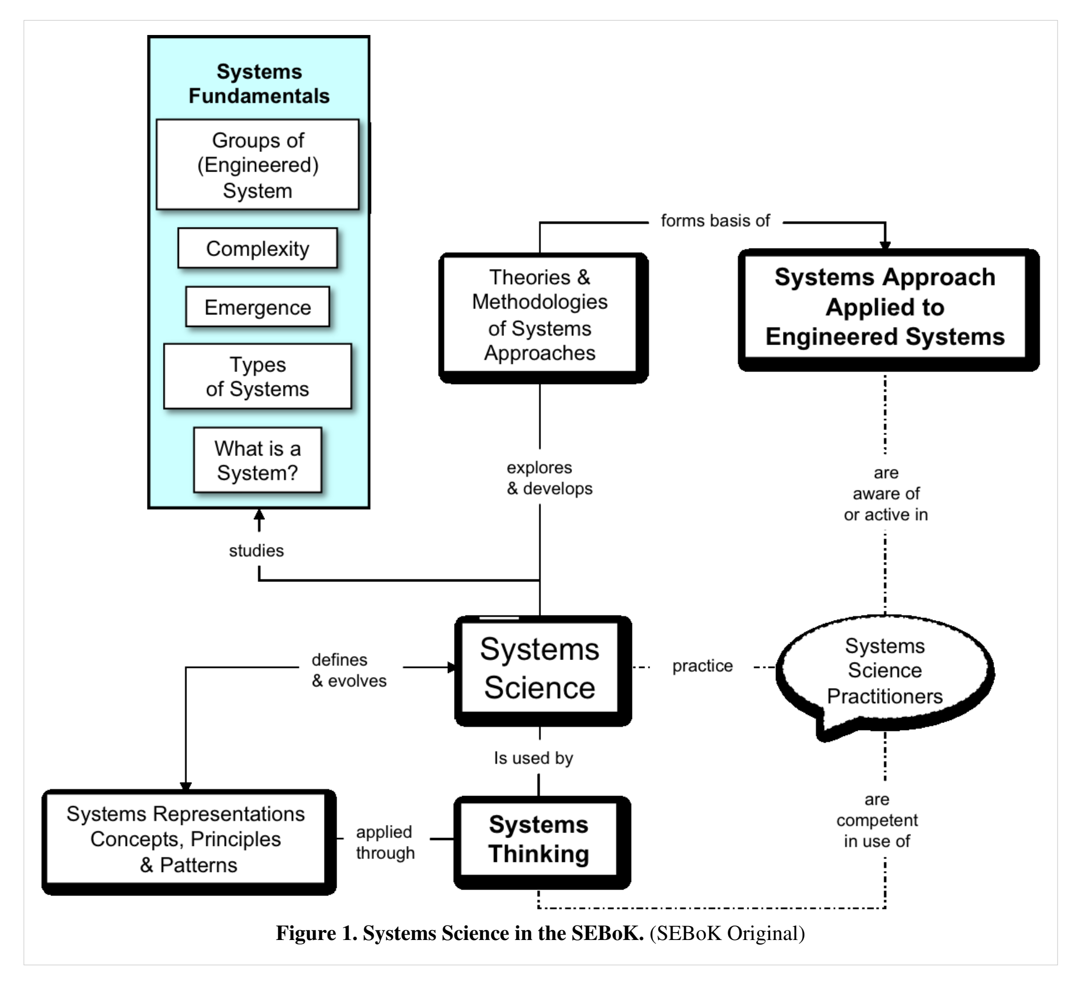


Figure 1. Systems Science in the SEBoK. (SEBoK Original)

Systems science brings together research into all aspects of systems with the goal of identifying, exploring, and understanding patterns of complexity which cross disciplinary fields and areas of application. It seeks to develop interdisciplinary foundations which can form the basis of theories applicable to all types of systems, independent of element type or application; additionally, it could form the foundations of a meta-discipline unifying traditional scientific specialisms.

The History of Systems Science article describes some of the important multidisciplinary fields of research of which systems science is composed.

A second article presents and contrasts the underlying theories behind some of the system approaches taken in applying systems science to real problems.

People who think and act in a systems way are essential to the success of both research and practice. Successful systems research will not only apply systems thinking to the topic being researched but should also consider a systems thinking approach to the way the research is planned and conducted. It would also be of benefit to have people involved in research who have, at a minimum, an awareness of system practice and ideally are involved in practical applications of the theories they develop.

References

Works Cited

None.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Flood, R.L. 1999. *Rethinking the Fifth Discipline: Learning within the Unknowable.* London, UK: Routledge.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 12 October 2018

History of Systems Science

This article is part of the Systems Science knowledge area (KA). It describes some of the important multidisciplinary fields of research comprising systems science in historical context.

Systems science, is an integrative discipline which brings together ideas from a wide range of sources which share a common systems theme. Some fundamental concepts now used in systems science have been present in other disciplines for many centuries, while equally fundamental concepts have independently emerged as recently as 40 years ago (Flood and Carson 1993).

The "Systems Problem"

Questions about the nature of systems, organization, and complexity are not specific to the modern age. As International Council on Systems Engineering (INCOSE) pioneer and former International Society for System Sciences (ISSS) President John Warfield put it, "Virtually every important concept that backs up the key ideas emergent in systems literature is found in ancient literature and in the centuries that follow." (Warfield 2006) It was not until around the middle of the 20th Century, however, that there was a growing sense of a need for, and possibility of a scientific approach to problems of organization and complexity in a "science of systems" per se.

The explosion of knowledge in the natural and physical sciences during the 18th and 19th centuries had made the creation of specialist disciplines inevitable: in order for science to advance, there was a need for scientists to become expert in a narrow field of study. The creation of educational structures to pass on this knowledge to the next generation of specialists perpetuated the fragmentation of knowledge (M'Pherson 1973).

This increasing specialization of knowledge and education proved to be a strength rather than a weakness for problems which were suited to the prevailing scientific methods of experimental isolation and analytic reduction. However there were areas of both basic and applied science that were not adequately served by those methods alone. The systems movement has its roots in two such areas of science: the biological-social sciences, and a mathematical-managerial base stemming first from cybernetics and operations research, and later from organizational theory.

Biologist Ludwig von Bertalanffy was one of the first to argue for and develop a broadly applicable scientific research approach based on **Open System Theory** (Bertalanffy 1950). He explained the scientific need for systems research in terms of the limitations of analytical procedures in science.

These limitations, often expressed as emergent evolution or "the whole is more than a sum of its parts", are based on the idea that an entity can be resolved into and reconstituted from its parts, either material or conceptual:

> *This is the basic principle of "classical" science, which can be circumscribed in different ways: resolution into isolable causal trains or seeking for "atomic" units in the various fields of science, etc.*

He stated that while the progress of "classical" science has shown that these principles, first enunciated by Galileo and Descartes, are highly successful in a wide realm of phenomena, but two conditions are required for these principles to apply:

> *The first is that interactions between "parts" be non-existent or weak enough to be neglected for certain research purposes. Only under this condition, can the parts be "worked out," actually, logically, and mathematically, and then be "put together." The second condition is that the relations describing the behavior of parts be linear; only then is the condition of summativity given, i.e., an equation describing the behavior of the total is of the same form as the equations describing the behavior of the parts.*
>
> *These conditions are not fulfilled in the entities called systems, i.e. consisting of parts "in interaction" and description by nonlinear mathematics. These system entities describe many real world situations: populations, eco systems, organizations and complex man made technologies. The methodological problem of systems theory is to provide for problems beyond the analytical-summative ones of classical science.* (Bertalanffy 1968, 18-19)

Bertalanffy also cited a similar argument by mathematician and co-founder of information theory Warren Weaver in a 1948 American Scientist article on "Science and Complexity". Weaver had served as Chief of the Applied Mathematics Panel at the U.S. Office of Scientific Research and Development during WWII. Based on those experiences, he proposed an agenda for what he termed a new "science of problems of organized complexity".

Weaver explained how the mathematical methods which had led to great successes of science to date were limited to problems where appropriate simplifying assumptions could be made. What he termed "problems of simplicity" could be adequately addressed by the mathematics of mechanics, while "problems of disorganized complexity" could be successfully addressed by the mathematics of statistical mechanics. But with other problems, making the simplifying assumptions in order to use the methods would not lead to helpful solutions. Weaver placed in this category problems such as, how the genetic constitution of an organism expresses itself in the characteristics of the adult, and to what extent it is safe to rely on the free interplay of market forces if one wants to avoid wide swings from prosperity to depression. He noted that these were complex problems which involved "analyzing systems which are organic wholes, with their parts in close interrelation."

> *These problems-and a wide range of similar problems in the biological, medical, psychological, economic, and political sciences-are just too complicated to yield to the old nineteenth century techniques which were so dramatically successful on two-, three-, or four-variable problems of simplicity. These new problems, moreover, cannot be handled with the statistical techniques so effective in describing average behavior in problems of disorganized complexity [problems with elements exhibiting random or unpredictable behaviour].*

These new critical global problems require science to make a third great advance,

> *An advance that must be even greater than the nineteenth-century conquest of problems of simplicity or the twentieth-century victory over problems of disorganized complexity. Science must, over the next 50 years, learn to deal with these problems of organized complexity [problems for which complexity "emerges" from the coordinated interaction between its parts.* (Weaver 1948)

Weaver identified two grounds for optimism in taking on this great challenge: 1.) developments in mathematical modeling and digital simulation, and 2.) the success during WWII of the “mixed team” approach of operations analysis, where individuals from across disciplines brought their skills and insights together to solve critical, complex problems.

The importance of modeling and simulation and the importance of working across disciplinary boundaries have been the key recurring themes in development of this “third way” science for systems problems of organized complexity.

The Development of Systems Research

The following overview of the evolution of systems science is broadly chronological, but also follows the evolution of different paradigms in system theory.

Open Systems and General Systems Theory

General system theory (GST) attempts to formulate principles relevant to all open systems (Bertalanffy 1968). GST is based on the idea that correspondence relationships (homologies) exist between systems from different disciplines. Thus, knowledge about one system should allow us to reason about other systems. Many of the generic system concepts come from the investigation of GST.

In 1954, Bertalanffy co-founded, along with Kenneth Boulding (economist), Ralph Gerard (physiologist) and 'Anatol Rapoport (mathematician), the Society for General System Theory (renamed in 1956 the Society for General Systems Research, and in 1988 the ISSS.

The initial purpose of the society was "to *encourage the development of theoretical systems which are applicable to more than one of the traditional departments of knowledge ... and promote the unity of science through improving the communication among specialists*." (Bertalanffy 1968)

This group is considered by many to be the founders of **System Age Thinking** (Flood 1999).

Cybernetics

Cybernetics was defined by Wiener, Ashby and others as the study and modeling of communication, regulation, and control in systems (Ashby 1956; Wiener 1948). Cybernetics studies the flow of information through a system and how information is used by the system to control itself through feedback mechanisms. Early work in cybernetics in the 1940s was applied to electronic and mechanical networks, and was one of the disciplines used in the formation of early systems theory. It has since been used as a set of founding principles for all of the significant system disciplines.

Some of the key concepts of feedback and control from Cybernetics are expanded in the Concepts of Systems Thinking article.

Operations Research

Operations Research (OR) considers the use of technology by an organization. It is based on the use of mathematical modeling and statistical analysis to optimize decisions on the deployment of the resources under an organization's control. An interdisciplinary approach based on scientific methods, OR arose from military planning techniques developed during World War II.

Operations Research and Management Science (ORMS) was formalized in 1950 by Ackoff and Churchman applying the ideas and techniques of OR to organizations and organizational decisions (Churchman et. al. 1950).

Systems Analysis

Systems analysis was developed by RAND Corporation in 1948. It borrowed from and extended OR, including using black boxes and feedback loops from cybernetics to construct block diagrams and flow graphs. In 1961, the Kennedy Administration decreed that systems analysis techniques should be used throughout the government to provide a quantitative basis for broad decision-making problems, combining OR with cost analysis. (Ryan 2008)

Systems Dynamics

Systems dynamics (SD) uses some of the ideas of cybernetics to consider the behavior of systems as a whole in their environment. SD was developed by Jay Forrester in the 1960's (Forrester 1961). He was interested in modeling the dynamic behavior of systems such as populations in cities, industrial supply chains. See Systems Approaches for more details.

SD is also used by Senge (Senge 1990) in his influential book *The Fifth Discipline*. This book advocates a system thinking approach to organization and also makes extensive use of SD notions of feedback and control.

Organizational Cybernetics

Stafford Beer was one of the first to take a cybernetics approach to organizations (Beer 1959). For Beer the techniques of ORMS are best applied in the context of an understanding of the whole system. Beer also developed a **Viable Systems Model** (Beer 1979), which encapsulates the effective organization needed for a system to be viable (to survive and adapt in its environment).

Work in cybernetics and ORMS consider the mechanism for communication and control in complex systems, and particularly in organizations and management sciences. They provide useful approaches for dealing with operational and tactical problems within a system, but do not allow consideration of more strategic organizational problems (Flood 1999).

Hard and Soft Systems Thinking

Action research is an approach, first described by Kurt Lewin, as a reflective process of progressive problem solving in which reflection on action leads to a deeper understanding of what is going on and to further investigation (Lewin 1958).

Peter Checkland's action research program in the 1980's led to an Interpretative-based Systemic Theory which seeks to understand organizations by not only observing the actions of people, but also by building understandings of the cultural context, intentions and perceptions of the individuals involved. Checkland, himself starting from a systems engineering (SE) perspective, successively observed the problems in applying a SE approach to the more fuzzy, ill-defined problems found in the social and political arenas (Checkland 1978). Thus he introduced a distinction between hard systems and soft systems - see also Systems Approaches.

Hard systems (glossary) views of the world are characterized by the ability to define purpose, goals, and missions that can be addressed via engineering methodologies in an attempt to, in some sense, "optimize" a solution.

In hard system approaches the problems may be complex and difficult, but they are known and can be fully expressed by the investigator. Such problems can be solved by selecting from the best available solutions (possibly with some modification or integration to create an optimum solution). In this context, the term "systems" is used to describe real world things; a solution system is selected, created and then deployed to solve the problem.

Soft systems (glossary) views of the world are characterized by complex, problematical, and often mysterious phenomena for which concrete goals cannot be established and which require learning in order to make improvement. Such systems are not limited to the social and political arenas and also exist within and amongst enterprises where complex, often ill-defined patterns of behavior are observed that are limiting the enterprise's ability to improve.

Soft system approaches reject the idea of a single problem and consider **problematic** situations in which different people will perceive different issues depending upon their own viewpoint and experience. These problematic situations are not solved, but managed through interventions which seek to reduce "discomfort" among the participants. The term system is used to describe systems of ideas, conceptual systems which guide our understanding of the situation, or help in the selection of intervention strategies.

These three ideas of “problem vs. problematic situation”,“solution vs. discomfort reduction”, and “the system vs. systems understanding” encapsulate the differences between hard and soft approaches (Flood and Carson 1993).

Critical Systems Thinking

The development of a range of hard and soft methods naturally leads to the question of which method to apply in what circumstances (Jackson 1989). Critical systems thinking (CST), or **critical management science** (Jackson 1985), attempts to deal with this question.

The word **critical** is used in two ways. Firstly, critical thinking considers the limits of knowledge and investigates the limits and assumptions of hard and soft systems, as discussed in the above sections. The second aspect of critical thinking considers the ethical, political and coercive dimension and the role of system thinking in society, see also Systems Approaches.

Service Science and Service Systems Engineering

The world economies have transitioned over the past few decades from manufacturing economies that provide goods - to service based economies. Harry Katzan defined the newly emerging field of service science: "Service science is defined as the application of scientific, engineering, and management competencies that a service-provider organization performs that creates value for the benefit of the client of customer" (Katzan 2008, vii).

The disciplines of service science and service engineering have developed to support this expansion and are built on principles of systems thinking but applied to the development and delivery of service systems.

Service Systems Engineering is described more fully in the Service Systems Engineering KA in Part 4 of the SEBoK.

References

Works Cited

Ackoff, R.L. 1971. "Towards a System of Systems Concepts". *Management Science.* 17(11).

Ashby, W. R. 1956. *Introduction to Cybernetics.* London, UK: Methuen.

Beer, S. 1959. *Cybernetics and Management.* London, UK: English Universities; New York: Wiley and Sons.

Beer, S. 1979. *The Heart of the Enterprise*. Chichester, UK: Wiley.

Bertalanffy, L. von. 1950. "The Theory of Open Systems in Physics and Biology". *Science*, New Series, 111(2872) (Jan 13): 23-29

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Checkland, P. 1978. "The Origins and Nature of “Hard” Systems Thinking." *Journal of Applied Systems Analysis*, 5(2): 99-110.

Checkland, P. 1999. *Systems Thinking, Systems Practice*, New York, NY, USA: John Wiley & Sons.

Churchman, C.W. 1968. *The Systems Approach.* New York, NY, USA: Dell Publishing.

Churchman, C.W., R.L. Ackoff. and E.L. Arnoff. 1950. *Introduction to Operations Research.* New York, NY, USA: Wiley and Sons.

Flood, R.L. 1999. *Rethinking the Fifth Discipline: Learning within the Unknowable.* London, UK: Routledge.

Flood, R.L. and E.R. Carson. 1993. *Dealing with Complexity: An Introduction to the Theory and Application of Systems Science,* 2nd ed. New York, NY, USA: Plenum Press.

Forrester, J. 1961. *Industrial Dynamics*. Cambridge, MA, USA: MIT Press.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Jackson, M. 1989. "Which Systems Methodology When? Initial Results from a Research Program." In: R Flood, M Jackson and P Keys (eds). *Systems Prospects: the Next Ten Years of Systems Research.* New York, NY, USA: Plenum.

Jackson, M. 2003. *Systems Thinking: Creating Holisms for Managers*. Chichester, UK: Wiley.

Katzan, H. 2008. *Service Science.* Bloomington, IN, USA: iUniverse Books.

Lewin, K. 1958. *Group Decision and Social Change.* New York, NY, USA: Holt, Rinehart and Winston. p. 201.

Magee, C. L., O.L. de Weck. 2004. "Complex System Classification." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June 2004, Toulouse, France.

M'Pherson, P, K. 1974. "A Perspective on Systems Science and Systems Philosophy." *Futures* 6(3) (June 1974): 219-239.

Miller, J.G. 1986. "Can Systems Theory Generate Testable Hypothesis?: From Talcott Parsons to Living Systems Theory." *Systems Research.* 3: 73-84.

Ryan, A. 2008. “What is a Systems Approach?” *Journal of Nonlinear Science.*

Senge, P.M. 1990. *The fifth discipline: The Art & Practice of the Learning Organization.* New York, NY, USA: Doubleday Business.

Weaver, W. (1948). “Science and complexity.” *American Scientist*. 36: 536-544.

Wiener, N. 1948. *Cybernetics or Control and Communication in the Animal and the Machine.* Paris, France: Hermann & Cie Editeurs; Cambridge, MA, USA: The Technology Press; New York, NY, USA: John Wiley & Sons Inc.

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and Sons.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Flood, R. L. 1999. *Rethinking the Fifth Discipline: Learning within the Unknowable.* London, UK: Routledge.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Additional References

Ackoff, R.L. 1981. *Creating the Corporate Future.* New York, NY, USA: Wiley and Sons.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Bowler, D.T. 1981. *General Systems Thinking: Its Scope and Applicability.* Amsterdam: The Netherlands: Elsevier.

Boulding, K.E. 1996. *The World as a Total System.* Beverly Hills, CA, USA: Sage Publications.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* Hoboken, NJ, USA: Wiley.

Laszlo, E. (ed). 1972. *The Relevance of General Systems Theory.* New York, NY, USA: George Brazillier.

Skyttner, L. 1996. *General Systems Theory - An Introduction.* Basingstoke, UK: Macmillan Press.

Warfield, J.N. 2006. *An Introduction to Systems Science.* Singapore: World Scientific Publishing Co. Pte Ltd. Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and Sons.

Lusch, R.F. and S. L. Vargo (Eds). 2006. *The service-dominant logic of marketing: Dialog, debate, and directions.* Armonk, NY: ME Sharpe Inc.

Maglio P., S. Srinivasan, J.T. Kreulen, and J. Spohrer. 2006. “Service Systems, Service Scientists, SSME, and Innovation." *Communications of the ACM*. 49(7) (July).

Popper, K. R. 1979. *Objective Knowledge*, 2nd edition. Oxford, UK: Oxford University Press.

Salvendy, G. and W. Karwowski (eds.). 2010. *Introduction to Service Engineering*. Hoboken, NJ, USA: John Wiley and Sons.

Sampson, S.E. 2001. *Understanding Service Businesses*. New York, NY, USA: John Wiley.

Spohrer, J. and P. P. Maglio. 2008. "The emergence of service science: Toward systematic service innovations to accelerate co-creation of value." *Production and Operations Management* 17 (3): 238-246, cited by Spohrer, J. and P. Maglio. 2010. "Service Science: Toward a Smarter Planet." In *Introduction to Service Engineering*. Ed. G Salvendy and W Karwowski. 3-30. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Systems Approaches

This article is part of the Systems Science knowledge area (KA). It presents issues in the comparison and analysis of systems approaches by the systems science community. Some of these ideas contribute to basic theory and methods that are used in systems thinking discussed in the Systems Thinking KA.

What is a Systems Approach?

In Bertalanffy's introduction to his 1968 book *General System Theory* (GST), he characterizes a systems approach as:

> *A certain objective is given; to find ways and means for its realization requires the system specialist (or team of specialists) to consider alternative solutions and to choose those promising optimization at maximum efficiency and minimum cost in a tremendously complex network of interactions.* (Bertalanffy 1968, 4)

He goes on to list as possible elements of a systems approach: “classical” systems theory (differential equations), computerization and simulation, compartment theory, set theory, graph theory, net theory, cybernetics, information theory, theory of automata, game theory, decision theory, queuing theory, and models in ordinary language.

This description is similar to what Warren Weaver identified as the methods used successfully by “mixed teams” during World War II (WWII) on “problems of organized complexity”. However, some conditions that had contributed to success during wartime did not hold after the war, such as a clear focus on well-defined common goals that motivated participants to work across disciplinary boundaries.

By the early 1970’s, there was growing disillusionment with the promise that a systems approach would provide easy solutions for all complex problems. There was particular criticism from some, including pioneers of Operations Research and Management Science (ORMS) like Ackoff and Churchman, that reliance on rote mathematical methods to identify optimal solutions among fixed alternatives had become just as inflexible and unimaginative an approach to complex problems as whatever it had replaced. Interest grew in examining and comparing methods and methodologies to better understand what could help ensure the best thinking and learning in terms of systems in systems approaches to practice.

Issues in Systems Approaches

A systems approach is strongly associated with systems thinking and how it helps to guides systems practice. In What is Systems Thinking? the key ideas of considering a system holistically, setting a boundary for a problem/solution of interest, and considering the resulting system-of-interest from outside its boundary are identified (Churchman 1979; Senge 2006).

A systems approach can view a system as a “holon” – an entity that is itself a “whole system” that interacts with a mosaic of other holons in its wider environment (Hybertson 2009), while also being made up of interacting parts. We can use this model recursively – each part of the system may be a system in its own right, and can itself be viewed both as an entity as seen from outside, and as a set of interacting parts. This model also applies in upwards recursion, so the original “system-of-interest” is an interacting part of one or more wider systems.

This means that an important skill in a systems approach is to identify the “natural holons” in the problem situation and solution systems and to make the partitioning of responsibilities match the “natural holons”, so as to minimize the coupling between parallel activities when applying a solution. This is the “cohesive/loose coupling” heuristic that has been around for a long time in many design disciplines.

Another consequence of the holistic nature of a systems approach is that it considers not only a problem situation and a solution system but also the system created and deployed to apply one to the other. A systems approach must

consider both the boundary of the system of concern as well as the boundary of the system inquiry (or model). Real systems are always open, i.e., they interact with their environment or supersystem(s). On the other hand, real models are always “closed” due to resource constraints — a fixed boundary of consideration must be set. So there is an ongoing negotiation to relate the two in systems practice and the judgment to do so is greatly helped by an appreciation of the difference between them.

Thus, a systems approach can be characterized by how it considers problems, solutions and the problem resolution process itself:

- Consider problems holistically, setting problem boundaries though understanding of natural system relationships and trying to avoid unwanted consequences.
- Create solutions based on sound system principles, in particular creating system structures which reduce organized complexity and unwanted emergent properties.
- Use understanding, judgment and models in both problem understanding and solution creation, while understanding the limitations of such views and models.

Systems Methodologies

One topic that has received significant attention in the systems science community is the analysis and comparison of methodologies which implement a systems approach. A methodology is a body of tools, procedures, and methods applied to a problem situation, ideally derived from a theoretical framework. These describe structured approaches to problem understanding and/or resolution making use of some of the concepts of systems thinking. These methodologies are generally associated with a particular system paradigm or way of thinking, which has a strong influence on the three aspects of a systems approach described above.

The most widely used groups of methodologies are as follows, see also History of Systems Science:

- Hard System (glossary) methodologies (Checkland 1978) set out to select an efficient means to achieve a predefined and agreed end.
- Soft System (glossary) methodologies (Checkland 1999) are interactive and participatory approaches to assist groups of diverse participants to alleviate a complex, problematic situation of common interest.
- Critical Systems Thinking (glossary) methodologies (Jackson 1985) attempts to provide a framework in which appropriate hard and soft methods can be applied as appropriate to the situation under investigation.

Systems Dynamics

Systems dynamics (SD) uses some of the ideas of cybernetics to consider the behavior of systems as a whole in their environment. SD was developed by Jay Forrester in the 1960’s. He was interested in modeling the dynamic behavior of systems such as populations in cities, or industrial supply chains.

System dynamics, (Forrester 1961), is an approach to understanding the behavior of complex systems over time. It deals with internal feedback loops and time delays that affect the behavior of the entire system. The main elements of SD are:

- The understanding of the dynamic interactions in a problem or solution as a system of feedback loops, modeled using a Causal Loop Diagram.
- Quantitative modeling of system performance as an accumulation of stocks (any entity or property which varies over time) and flows (representations of the rate of change of a stock).
- The creation of dynamic simulations, exploring how the value of key parameters change over time. A wide range of software tools are available to support this.

These elements help describe how even seemingly simple systems display baffling non-linearity.

Hard Systems Methodologies

Checkland (Checkland 1975) classifies hard system (glossary) methodologies, which set out to select an efficient means to achieve a predefined end, under the following headings:

- System Analysis (glossary) - the systematic appraisal of the costs and other implications of meeting a defined requirement in various ways.
- Systems Engineering (glossary) (SE) - the set of activities that together lead to the creation of a complex man-made entity and/or the procedures and information flows associated with its operation.

Operational Research is also considered a hard system approach, closely related to the systems analysis approach developed by the Rand Corporation, in which solutions are known but the best combinations of these solutions must be found. There is some debate as to whether system dynamics is a hard approach, which is used to assess the objective behavior of real situations. Many application of SD have focused on the system, however it can and has also be used as part of a soft approach including the modeling of subjective perceptions (Lane 2000).

SE allows for the creation of new solution systems, based upon available technologies. This hard view of SE as a solution focused approach applied to large, complex and technology focused solutions, is exemplified by (Jenkins 1969; Hall 1962) and early defense and aerospace standards.

It should be noted that historically the SE discipline was primarily aimed at developing, modifying or supporting hard systems. More recent developments in SE have incorporated problem focused thinking and agile solution approaches. It is this view of SE that is described in the SEBoK.

All of these hard approaches can use systems thinking to ensure complete and viable solutions are created and/or as part of the solution optimization process. These approaches are appropriate to unitary problems, but not when the problem situation or solution technologies are unclear.

Soft Systems and Problem Structured Methods

Problem Structuring Methods (PSM) are interactive and participatory approaches to assist groups of diverse participants to alleviate a complex, problematic situation of common interest. Typically the hardest element of the situation is framing the issues which constitute the problem (Minger and Resenhead 2004).

PSM use systems and systems thinking as an abstract framework for investigation, rather than a structure for creating solutions. Systems descriptions are used to understand the current situation and describe an idealized future. Interventions directly in the current organization to move towards the idea recognize that the assumptions and mental models of the participants are an important obstruction to change and that these differing views cannot be dismissed, but instead must form part of the intervention approach.

Peter Checkland's action research program, see Systems Science, in the 1980's forms the basis of work by Checkland, Wilson and others in the development of Soft Systems Methodology (glossary) (SSM) (Checkland 1999; Wilson 2001). SSM formalizes the idea of a soft approach using systemic thinking to expose the issues in a problem situation and guide interventions to reduce them. SSM provides a framework of ideas and models to help guide participants through this systemic thinking.

Other PSM approaches include interactive planning approach (Ackoff 1981), social systems design (Churchman 1968), and strategic assumptions surfacing and testing (Mason and Mitroff 1981).

SSM and other soft approaches use systems thinking to ensure problem situations are fully explored and resolved. These approaches are appropriate to pluralist (glossary) problems. Critics of SSM suggest that it does not consider the process of intervention, and in particular how differences in power between individuals and social groups impact the effectiveness of interventions.

Critical Systems Thinking and Multi-Methodology

The development of a range of hard and soft methods naturally leads to the question of which method to apply in what set of circumstances (Jackson 1989). Critical systems thinking (CST) or **Critical Management Science** Jackson (Jackson 1985) attempts to deal with this question.

The word critical is used in two ways. Firstly, critical thinking considers the limits of knowledge and investigates the limits and assumptions of hard and soft systems, as discussed in the above sections. From this comes frameworks and meta-methodology that establish when to apply different methods such as **total systems intervention** (TSI) (Flood and Jackson 1991). Critical or “pluralist” or “pragmatic” , **multi-methodology** approaches take this aspect of critical thinking one stage further to recognize the value of combining techniques from several hard, soft , or custom methods as needed (Mingers and Gill 1997). Many in the systems science community believe that the multi-methodology approach has been accepted as the de facto systems approach and that the challenges now are in refining tools and methods to support it.

Churchman (Churchman, 1979) and others have also considered broader ethics political and social questions related to management science, with regards to the relative power and responsibility of the participants in system interventions. The second aspect of critical thinking considers the ethical, political, and coercive dimension in Jackson's System of Systems Methodologies (SOSM) framework (Jackson 2003) and the role of system thinking in society.

Selecting Systems Methodologies

Jackson proposes a frame for considering which approach should be applied, please see Jackson's Framework [1]. In Jackson's framework the following definitions apply to the participants involved in solving the problem:

- Unitary (glossary) - A problem situation in which participants "have similar values, beliefs and interests. They share common purposes and are all involved, in one way or another, in decision-making about how to realize their agreed objectives." (Jackson 2003, 19)
- Pluralist (glossary) - A problem situation involving participants in which "although their basic interests are compatible, they do not share the same values and beliefs. Space needs to be made available within which debate, disagreement, even conflict, can take place. If this is done, and all feel they have been involved in decision-making, then accommodations and compromises can be found. Participants will come to agree, at least temporarily, on productive ways forward and will act accordingly." (Jackson 2003, 19)
- Coercive (glossary) - A problem situation in which the participants "have few interests in common and, if free to express them, would hold conflicting values and beliefs. Compromise is not possible and so no agreed objectives direct action. Decisions are taken on the basis of who has most power and various forms of coercion employed to ensure adherence to commands." (Jackson 2003, 19)

Jackson's framework suggests that for simple and complex systems with unitary participants, hard and dynamic systems thinking applies, respectively. For simple and complex systems with pluralist participants, soft systems thinking applies. For simple and complex systems with coercive participants, **emancipatory** and postmodernist system thinking applies, respectively. These thinking approaches consider all attempts to look for system solutions to be temporary and ineffective in situations where the power of individuals and groups of people dominate any system structures we create. They advocate an approach which encourages diversity, free-thinking and creativity of individuals and in the organization's structures. Thus, modern system thinking has the breadth needed to deal with a broad range of complex problems and solutions.

These ideas sit at the extreme of system thinking as a tool for challenging assumptions in and stimulating innovative solutions in problem solving. Jackson (Jackson 2003) identifies the work of some authors who have included these ideas into their systems approach.

References

Works Cited

Bertalanffy, L. 1968. *General System Theory: Foundations, Development, Applications*. New York, NY, USA: George Braziller, Inc.

Ackoff, R.L. 1981. *Creating the Corporate Future*. New York, NY, USA: Wiley and Sons.

Checkland, P. 1978. "The Origins and Nature of “Hard” Systems Thinking." *Journal of Applied Systems Analysis.* 5(2): 99-110.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Churchman, C.W. 1968. *The Systems Approach*. New York, NY, USA: Dell Publishing.

Churchman, C. West. 1979. *The Systems Approach and Its Enemies*. New York: Basic Books.

Flood, R. and M. Jackson. 1991. *Creative Problem Solving: Total Systems Intervention*. London, UK: Wiley.

Forrester, J. 1961. *Industrial Dynamics*. Cambridge, MA, USA: MIT Press.

Hall, A.D. 1962. *A Methodology for Systems Engineering*. New York, NY, USA: Van Nostrand Reinhold.

Hybertson, D, 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems.* Series in Complex and Enterprise Systems Engineering. Boston, MA, USA: Auerbach Publications.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Jackson, M. 1989. "Which Systems Methodology When? Initial Results from a Research Program." In: R Flood, M Jackson and P Keys (eds). *Systems Prospects: the Next Ten Years of Systems Research*. New York, NY, USA: Plenum.

Jackson, M. 2003. *Systems Thinking: Creating Holisms for Managers*. Chichester, UK: Wiley.

Jenkins, G.M. 1969. "The Systems Approach." In Beishon, J. and G. Peters (eds.), *Systems Behavior*, 2nd ed. New York, NY, USA: Harper and Row.

Lane, D. 2000. “Should System Dynamics be Described as a `Hard' or `Deterministic' Systems Approach?” *Systems Research and Behavioral Science.* 17: 3–22 (2000).

Mason, R.O. and I.I. Mitroff. 1981. *Challenging Strategic Planning Assumptions: Theory, Case and Techniques*. New York, NY, USA: Wiley and Sons.

Mingers, J. and A. Gill. 1997. *Multimethodology: Theory and Practice of Combining Management Science Methodologies*. Chichester, UK: Wiley.

Senge, P.M. 1990, 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York, Doubleday/Currency.

Wilson, B. 2001. *Soft Systems Methodology—Conceptual Model Building and Its Contribution*. New York, NY, USA: J.H.Wiley.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Forrester, J. 1961. *Industrial Dynamics*. Cambridge, MA, USA: MIT Press.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Additional References

Jackson, M.C. and P. Keys. 1984. "Towards a System of Systems Methodologies." *The Journal of the Operational Research Society.* 35(6) (Jun. 1984): 473-486.

Mingers, J. and J. Rosenhead. 2004. "Problem Structuring Methods in Action." *European Journal of Operations Research.* 152(3) (Feb. 2004): 530-554. Sterman, J.D. 2001. "System dynamics modeling: Tools for learning in a complex world." *California Management Review.* 43(4): 8–25.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.systemswiki.org/index.php?title=System_of_Systems_Methodologies_(SOSM)

Knowledge Area: Systems Thinking

Systems Thinking

This knowledge area (KA) provides a guide to knowledge about systems thinking which is the integrating paradigm for systems science and systems approaches to practice.

This is part of the wider systems knowledge which can help to provide a common language and intellectual foundation; and make practical systems concepts, principles, patterns and tools accessible to systems engineering (SE), as discussed in the Introduction to Part 2.

Topics

Each part of the Guide to the SE Body of Knowledge (SEBoK) is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- What is Systems Thinking?
- Concepts of Systems Thinking
- Principles of Systems Thinking
- Patterns of Systems Thinking

Introduction

Systems thinking is concerned with understanding or intervening in problem situations, based on the principles and concepts of the systems paradigm. This KA offers some basic definitions of systems thinking. The following diagram summarizes how the knowledge is presented.

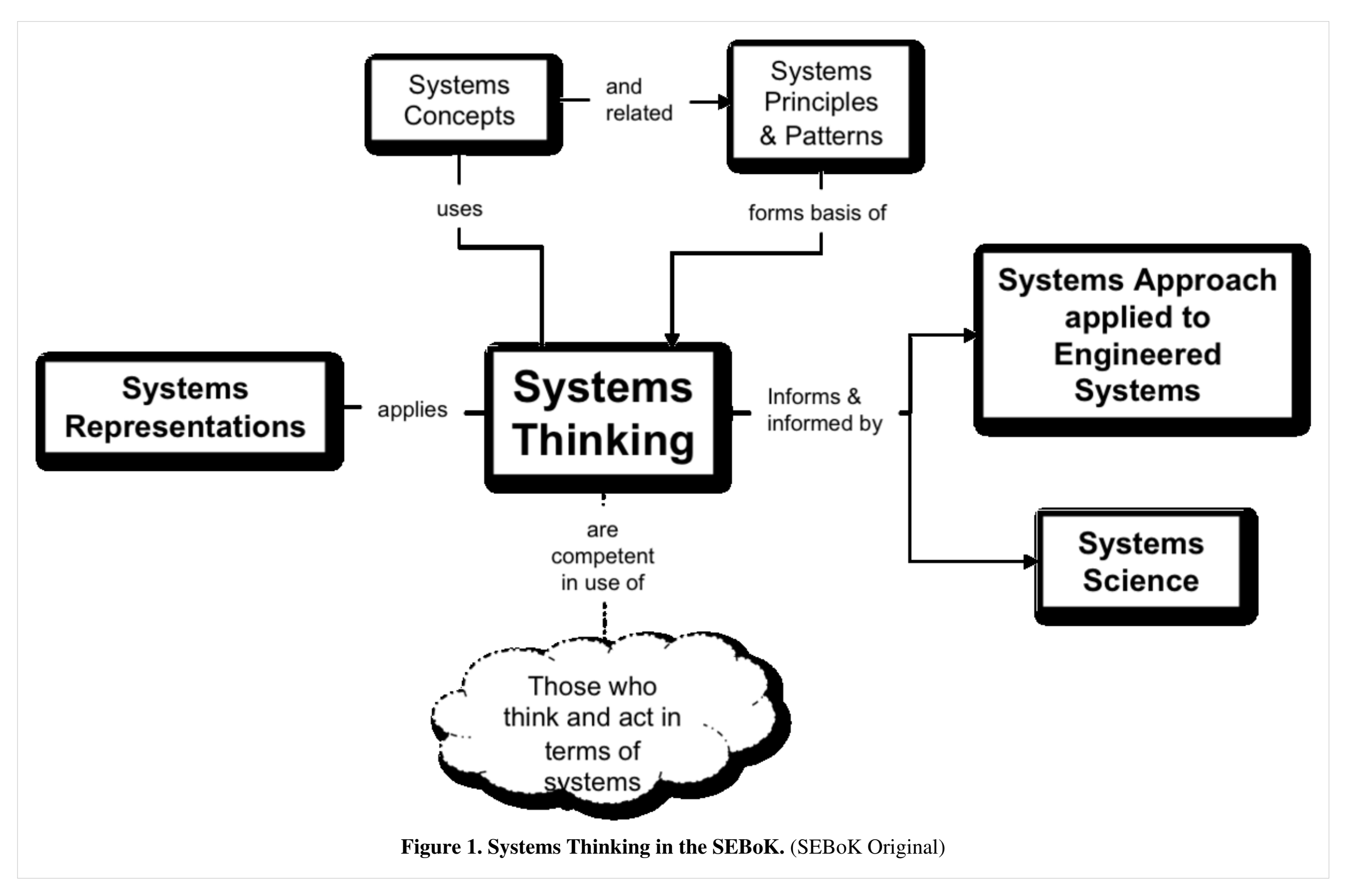


Figure 1. Systems Thinking in the SEBoK. (SEBoK Original)

Systems thinking considers the similarities between systems from different domains in terms of a set of common systems concepts, principles and patterns:

- A principle is a rule of conduct or behavior. To take this further, a principle is a “basic generalization that is accepted as true and that can be used as a basis for reasoning or conduct” (WordWeb.com).
- A concept is an abstraction, or a general idea inferred or derived from specific instances.

Principles depend on concepts in order to state a “truth.” Hence, principles and concepts go hand in hand; principles cannot exist without concepts and concepts are not very useful without principles to help guide the proper way to act (Lawson and Martin 2008).

Many sources combine both concepts and the principles based on them. The Concepts of Systems Thinking article presents concepts extracted from a variety of theory and practice sources. The Principles of Systems Thinking article, in turn, presents a summary of important principles referring back to the concepts upon which they are based is also provided.

A pattern is an expression of observable similarities found in systems from different domains. Patterns exist in both natural and man-made systems and are used in systems science and systems engineering. A summary of the different classes of patterns and the use of patterns to support a systems approach is discussed in the final Patterns of Systems Thinking article.

The practical application of systems thinking often employs the use of abstract system representations or models. Some mention of models is made in this KA; additionally, a more complete guide provided in Representing Systems with Models.

References

Works Cited

Lawson, H., and J.N. Martin. 2008. "On the Use of Concepts and Principles for Improving Systems Engineering Practice". Proceedings of the 18th Annual International Council on Systems Engineering (INCOSE) International Symposium, 5-19 June 2008, Utrecht, The Netherlands.

WordWeb Online. n.d. "Definition:Principle." Accessed Dec 3 2014 At WordWeb Online http:/ / www. wordwebonline.com/en/PRINCIPLE

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY: Braziller.

Checkland, P. 1999. *Systems Thinking, Systems Practice*, New York, NY, USA: John Wiley & Sons.

Churchman, C. W. 1968. *The Systems Approach and its Enemies*. New York, NY, USA: Dell Publishing.

Flood, R. L. 1999. *Rethinking the Fifth Discipline: Learning Within The Unknowable.* London UK: Routledge.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

Additional References

Ackoff, R.L. 1971. "Towards a System of Systems Concepts". *Management Science.* 17(11).

Hitchins, D. 2009. "What Are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Ramage, Magnus, and Karen Shipp. 2009. *Systems Thinkers*. Dordrecht: Springer.

Weinberg, Gerald M. 1975. *An Introduction to General Systems Thinking*. New York: Wiley.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

What is Systems Thinking?

This topic is part of the Systems Thinking knowledge area (KA). The scope of systems thinking is a starting point for dealing with real world situations using a set of related systems concept discussed in Concepts of Systems Thinking topic, systems principles discussed in Principles of Systems Thinking topic, and system patterns discussed in Patterns of Systems Thinking topic.

Introduction

The concepts, principles, and patterns of systems thinking have arisen both from the work of systems scientists and from the work of practitioners applying the insights of systems science to real-world problems.

Holism has been a dominant theme in systems thinking for nearly a century, in recognition of the need to consider a system as a whole because of observed phenomena such as emergence. Proponents have included Wertheimer, Smuts, Bertalanffy, Weiss, (Ackoff 1979), (Klir 2001), and (Koestler 1967) among many others.

A more detailed discussion of the most important movements in systems theory can be found in History of Systems Science.

Identifying Systems of Interest

When humans observe or interact with a system, they allocate boundaries and names to parts of the system. This naming may follow the natural hierarchy of the system, but will also reflect the needs and experience of the observer to associate elements with common attributes of purposes relevant to their own. Thus, a number of systems of interest (SoIs) (Flood and Carson 1993) must be identified and they must be both relevant and include a set of elements which represent a system whole. This way of observing systems wherein the complex system relationships are focused around a particular system boundary is called **systemic resolution**.

Systems thinking requires an ongoing process of attention and adaptation to ensure that one has appropriately identified boundaries, dependencies, and relationships. Churchman (Churchman 1968) and others have also considered broader ethical, political, and social questions related to management science with regards to the relative power and responsibility of the participants in system interventions. These are seen by critical systems thinkers as key factors to be considered in defining problem system boundaries.

A system context can be used to define a SoI and to capture and agree on the important relationships between it, such as the systems it works directly with and the systems which influence it in some way. When this approach is used to focus on part of a larger system, a balance of reductionism and holism is applied. This balance sits at the heart of a systems approach. A systems context provides the tool for applying this balance, and is thus an essential part of any systems approach and hence, of systems engineering (SE) as well. Approaches for describing the context of the different types of engineered systems are discussed in Engineered System Context topic within the Systems Approach Applied to Engineered Systems KA.

Thoughts on Systems Thinking

Senge (1990) discusses systems thinking in a number of ways as

> *a discipline for seeing wholes ... a framework for seeing interrelationships rather than things ... a process of discovery and diagnosis ... and as a sensibility for the subtle interconnectedness that gives living systems their unique character.*(Senge 2006, 68-69)

Churchman came to define a systems approach as requiring consideration of a system from the viewpoint of those outside its boundary (Churchman 1979). There are many demonstrations that choosing too narrow a boundary, either in terms of scope or timeline, results in the problem of the moment being solved only at the expense of a similar or bigger problem being created somewhere else in space, community, or time (Senge 2006) and (Meadows 1977). This is the "shifting the burden" archetype described in Patterns of Systems Thinking topic.

Churchman believes that an important component of system knowledge comes from "others" or "enemies" outside the system; the systems approach begins when first you see the world through the eyes of another (Churchman 1968). In this famous phrase, Churchman suggests that people can step outside a system they are in and mentally try to consider it through the lenses of other people's values. Churchman (1979) identified four main enemies of the systems approach namely: politics, morality, religion and aesthetics.

To Churchman, the "enemies" of the systems approach provide a powerful way of learning about the systems approach, precisely because they enable the rational thinker to step outside the boundary of a system and to look at it. It means that systems thinkers are not necessarily just involved within a system but are essentially involved in reasoning and decisions "outside" of systems rationality.

Some additional perspectives on systems thinking definitions are as follows:

- "Systems thinking requires the consciousness of the fact that we deal with models of our reality and not with the reality itself." (Ossimitz 1997, 1)
- "…what is often called "systemic thinking" …is …a bundle of capabilities, and at the heart of it is the ability to apply our normal thought processes, our common sense, to the circumstances of a given situation." (Dörner 1996, 199)
- "Systems thinking provides a powerful way of taking account of causal connections that are distant in time and space." (Stacey 2000, 9)

Chaos and complexity theories have also impacted the development of systems thinking, including the treatment of such concepts as emergence. According to Gharajedaghi:

> *Systems thinking is the art of simplifying complexity. It is about seeing through chaos, managing interdependency, and understanding choice. We see the world as increasingly more complex and chaotic because we use inadequate concepts to explain it. When we understand something, we no longer see it as chaotic or complex.* (Gharajedaghi 1999, 283)

Kasser considers systems thinking to be one element in a wider system of holistic thinking. Kasser defines holistic thinking as follows: "...the combination of analysis [in the form of elaboration], systems thinking and critical thinking." (Kasser 2010)

Systems Thinking and the Guide to the Systems Engineering Body of Knowledge

From these discussions one can see systems thinking as both a set of founding ideas for the development of systems theories and practices and also as a pervasive way of thinking need by those developing and applying them.

The SEBoK is particularly focused on how systems thinking can support a systems approach to engineered systems.

In order to examine a SoI in more detail, to understand, use, or change them in some way, practitioners are faced with an apparent “systems thinking paradox”. One can only truly understand a system by considering all of its possible relationships and interactions, inside and outside of its boundary and in all possible future situations (of both system creation and life), but this makes it apparently impossible for people to understand a system or to predict all of the consequences of changes to it.

If this means that all possible system relationships and environmental conditions must be considered to fully understand the consequences of creating or changing a system, what useful work can be done?

In many ways this is the essence of all human endeavors, whether they are technical, managerial, social or political, the so called known knowns and unknown unknowns. The systems approach is a way of tackling real world problems and making use of the concepts, principle and patterns of systems thinking to enable systems to be engineered and used.

The systems principles of encapsulation and separation of concerns in Principles of Systems Thinking relate to this issue. Some of the detail of complex situations must be hidden to allow focus on changes to a system element. The impact must be considered of any changes that might be made across sufficient related system components to fit within the acceptable commercial and social risks that must be considered. Engineering and management disciplines deal with this by gathering as much knowledge as necessary to proceed at a risk level acceptable to the required need. The assessment of what is enough and how much risk to take can, to some extent, be codified with rules and regulations, and managed through processes and procedures; however, it is ultimately a combination of the skill and judgment of the individuals performing the work.

References

Works Cited

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY: Braziller.

Churchman, C.W. 1968. *The Systems Approach*. Delacorte Press.

Churchman, C.W. 1979. "The Systems Approach and Its Enemies". New York: Basic Books.

Checkland, P. 1981. *Systems Thinking, Systems Practice*. New York, NY, USA: Wiley.

Dorner, H., and A. Karpati. 2008. "Mentored innovation in teacher training using two virtual collaborative learning environments." In *Beyond knowledge: The legacy of competence--meaningful computer-based learning environments*., eds. J. Zumbach, N. Schwartz, T. Seufert and L. Kester. Vol. VIII. New York, NY: Springer.

Flood, R.L. and E.R. Carson. 1993. *Dealing with Complexity: An Introduction to the Theory and Application of Systems Science*, 2nd ed. New York, NY, USA: Plenum Press.

Gharajedaghi, J. 1999. *Systems Thinking: Managing Chaos and Complexity: A platform for designing.* 1st ed. Woburn, MA: Butterworth-Heinemann.

Jackson, M. 1989. "Which Systems Methodology When? Initial Results from a Research Program." In: R Flood, M Jackson and P Keys (eds). *Systems Prospects: the Next Ten Years of Systems Research.* New York, NY, USA: Plenum.

Kasser, J. 2010. "Holistic thinking and how it can produce innovative solutions to difficult problems." Paper presented at 7th Bi-annual European Systems Engineering Conference (EuSEC), 24-27 May 2010, Stockholm, Sweden.

Meadows, Donella H. et al. 1977. "Limits to Growth: A Report for the Club of Rome's Project on the Predicament of Mankind." New American Library, paperback, ISBN 0-451-13695-0; Universe Books, hardcover, 1972, ISBN 0-87663-222-3 (scarce).

Ossimitz, G. "The development of systems thinking skills using system dynamics modeling tools." in Universitat Klagenfurt [database online]. Klagenfurt, Austria, 1997 [cited November 12 2007]. Available from http://wwwu.uni-klu.ac.at/gossimit/sdyn/gdm_eng.htm.

Senge, P.M. 1990, 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization.* New York, NY, USA: Doubleday Currency.

Stacey, R.D., D. Griffin, and P. Shaw. 2000. *Complexity and management: Fad or radical challenge to systems thinking?* London, U.K.: Routledge.

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY: Braziller.

Churchman, C.W.. 1979. "The Systems Approach and its Enemies". New York: Basic Books.

Gharajedaghi, J. 1999. *Systems Thinking: Managing Chaos and Complexity: A platform for designing.* 1st ed. Woburn, MA: Butterworth-Heinemann.

Senge, P.M. 1990, 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization.* New York, NY, USA: Doubleday Currency.

Additional References

Jackson, M. 2003. *Systems Thinking: Creating Holisms for Managers.* Wiley; Chichester

Edson, R. 2008. *Systems Thinking. Applied. A Primer*. In: ASYST Institute (ed.). Arlington, VA: Analytic Services.

Klir, G. 2001. *Facets of Systems Science, 2nd ed*. New York: Kluwer Academic/Plenum Publishers.

Koestler, A. 1967. *The Ghost in the Machine. New York*, NY: Macmillan. Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, Kings College, UK.

MITRE. 2012. “Systems Engineering Guide.” Accessed September 11, 2012. Available at http://www.mitre.org/work/systems_engineering/guide.

Rebovich, G., Jr. 2005. "Systems Thinking for the Enterprise (New and Emerging Perspectives)," In *Volume 2 of Enterprise Systems Engineering Theory and Practice*. The MITRE Corporation.

Senge, P.M., A. Klieiner, C. Roberts, R.B. Ross, and B.J. Smith. 1994. *The Fifth Discipline Fieldbook: Strategies and tools for building a learning organization*. New York, NY: Crown Business.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Concepts of Systems Thinking

This article forms part of the Systems Thinking knowledge area (KA). It describes systems concepts, knowledge that can be used to understand problems and solutions to support systems thinking.

The concepts below have been synthesized from a number of sources, which are themselves summaries of concepts from other authors. Ackoff (1971) proposed a system of system concepts as part of general system theory (GST); Skyttner (2001) describes the main GST concepts from a number of systems science authors; Flood and Carlson (1993) give a description of concepts as an overview of systems thinking; Hitchins (2007) relates the concepts to systems engineering practice; and Lawson (2010) describes a system of system concepts where systems are categorized according to fundamental concepts, types, topologies, focus, complexity, and roles.

Wholeness and Interaction

A system is defined by a set of elements which exhibit sufficient cohesion, or "togetherness", to form a bounded whole (Hitchins 2007; Boardman and Sauser 2008).

According to Hitchins, interaction between elements is the "key" system concept (Hitchins 2009, 60). The focus on interactions and holism is a push-back against the perceived reductionist focus on parts and provides recognition that in complex systems, the interactions among parts is at least as important as the parts themselves.

An open system is defined by the interactions between system elements within a system boundary and by the interaction between system elements and other systems within an environment (see What is a System?). The remaining concepts below apply to open systems.

Regularity

Regularity is a uniformity or similarity that exists in multiple entities or at multiple times (Bertalanffy 1968). Regularities make science possible and engineering efficient and effective. Without regularities, we would be forced to consider every natural and artificial system problem and solution as unique. We would have no scientific laws, no categories or taxonomies, and each engineering effort would start from a clean slate.

Similarities and differences exist in any set or population. Every system problem or solution can be regarded as unique, but no problem/solution is in fact entirely unique. The nomothetic approach assumes regularities among entities and investigates what the regularities are. The idiographic approach assumes each entity is unique and investigates the unique qualities of entities, (Bertalanffy 1975).

A very large amount of regularity exists in both natural systems and engineered systems. Patterns of systems thinking capture and exploit that regularity.

State and Behavior

Any quality or property of a system element is called an attribute. The state of a system is a set of system attributes at a given time. A **system event** describes any change to the environment of a system, and hence its state:

- **Static** - A single state exists with no events.
- **Dynamic** - Multiple possible stable states exist.
- **Homeostatic** - System is static but its elements are dynamic. The system maintains its state by internal adjustments.

A stable state is one in which a system will remain until another event occurs.

State can be monitored using state variables, values of attributes which indicate the system state. The set of possible values of state variables over time is called the "'state space'". State variables are generally continuous, but can be

modeled using a finite state model (or, "state machine").

Ackoff (Ackoff 1971) considers "change" to be how a system is affected by events, and system behavior as the effect a system has upon its environment. A system can

- **react** to a request by turning on a light,
- **respond** to darkness by deciding to turn on the light
- **act** to turn on the lights at a fixed time, randomly or with discernible reasoning.

A stable system is one which has one or more stable states within an environment for a range of possible events:

- **Deterministic** systems have a one-to-one mapping of state variables to state space, allowing future states to be predicted from past states.
- **Non-Deterministic** systems have a many-to-many mapping of state variables; future state cannot be reliably predicted.

The relationship between determinism and system complexity, including the idea of chaotic systems, is further discussed in the Complexity article.

Survival Behavior

Systems often behave in a manner that allows them to sustain themselves in one or more alternative viable states. Many natural or social systems have this goal, either consciously or as a "self organizing" system, arising from the interaction between elements.

Entropy is the tendency of systems to move towards disorder or disorganization. In physics, entropy is used to describe how organized heat energy is “lost” into the random background energy of the surrounding environment (the 2nd Law of Thermodynamics). A similar effect can be seen in engineered systems. What happens to a building or garden left unused for any time? Entropy can be used as a metaphor for aging, skill fade, obsolescence, misuse, boredom, etc.

"Negentropy" describes the forces working in a system to hold off entropy. Homeostasis is the biological equivalent of this, describing behavior which maintains a "steady state" or "dynamic equilibrium". Examples in nature include human cells, which maintain the same function while replacing their physical content at regular intervals. Again, this can be used as a metaphor for the fight against entropy, e.g. training, discipline, maintenance, etc.

Hitchins (Hitchins 2007) describes the relationship between the viability of a system and the number of connections between its elements. Hitchins's concept of connected variety states that stability of a system increases with its connectivity (both internally and with its environment). (See variety.)

Goal Seeking Behavior

Some systems have reasons for existence beyond simple survival. Goal seeking is one of the defining characteristics of engineered systems:

- A **goal** is a specific outcome which a system can achieve in a specified time
- An **objective** is a longer term outcome which can be achieved through a series of goals.
- An **ideal** is an objective which cannot be achieved with any certainty, but for which progress towards the objective has value.

Systems may be single goal seeking (perform set tasks), multi-goal seeking (perform related tasks), or reflective (set goals to tackle objectives or ideas). There are two types of goal seeking systems:

- Purposive (glossary) systems have multiple goals with some shared outcome. Such a system can be used to provide pre-determined outcomes within an agreed time period. This system may have some freedom to choose how to achieve the goal. If it has memory it may develop processes describing the behaviors needed for defined goals. Most machines or software systems are purposive.

- Purposeful (glossary) systems are free to determine the goals needed to achieve an outcome. Such a system can be tasked to pursue objectives or ideals over a longer time through a series of goals. Humans and sufficiently complex machines are purposeful.

Control Behavior

Cybernetics, the science of control, defines two basic control mechanisms:

- **Negative feedback**, maintaining system state against a set objectives or levels.
- **Positive feedback**, forced growth or contraction to new levels.

One of the main concerns of cybernetics is the balance between stability and speed of response. A black-box system (glossary) view looks at the whole system. Control can only be achieved by carefully balancing inputs with outputs, which reduces speed of response. A white-box system (glossary) view considers the system elements and their relationships; control mechanisms can be imbedded into this structure to provide more responsive control and associated risks to stability.

Another useful control concept is that of a "meta-system", which sits over the system and is responsible for controlling its functions, either as a black-box or white-box. In this case, behavior arises from the combination of system and meta-system.

Control behavior is a trade between

- **Specialization**, the focus of system behavior to exploit particular features of its environment, and
- Flexibility (glossary), the ability of a system to adapt quickly to environmental change.

While some system elements may be optimized for either specialization, a temperature sensitive switch, flexibility, or an autonomous human controller, complex systems must strike a balance between the two for best results. This is an example of the concept of dualism, discussed in more detail in Principles of Systems Thinking.

Variety describes the number of different ways elements can be controlled, and is dependent on the different ways in which they can then be combined. The Law of Requisite Variety states that a control system must have at least as much variety as the system it is controlling (Ashby 1956).

Function

Ackoff defines function as outcomes which contribute to goals or objectives. To have a function, a system must be able to provide the outcome in two or more different ways. (This is called **equifinality**).

This view of function and behavior is common in systems science. In this paradigm, all system elements have behavior of some kind; however, to be capable of functioning in certain ways requires a certain richness of behaviors.

In most hard systems approaches, a set of functions are described from the problem statement, and then associated with one or more alternative element structures (Flood and Carson 1993). This process may be repeated until a system component (implementable combinations of function and structure) has been defined (Martin 1997). Here, function is defined as either a task or activity that must be performed to achieve a desired outcome or as a transformation of inputs to outputs. This transformation may be:

- **Synchronous**, a regular interaction with a closely related system, or
- **Asynchronous**, an irregular response to a demand from another system that often triggers a set response.

The behavior of the resulting system is then assessed as a combination of function and effectiveness. In this case behavior is seen as an external property of the system as a whole and is often described as analogous to human or organic behavior (Hitchins 2009).

Hierarchy, Emergence and Complexity

System behavior is related to combinations of element behaviors. Most systems exhibit **increasing variety**; i.e., they have behavior resulting from the combination of element behaviors. The term "synergy", or weak emergence, is used to describe the idea that the whole is greater than the sum of the parts. This is generally true; however, it is also possible to get **reducing variety**, in which the whole function is less than the sum of the parts, (Hitchins 2007).

Complexity frequently takes the form of hierarchies (glossary). Hierarchic systems have some common properties independent of their specific content, and they will evolve far more quickly than non-hierarchic systems of comparable size (Simon 1996). A natural system hierarchy is a consequence of wholeness, with strongly cohesive elements grouping together forming structures which reduce complexity and increase robustness (Simons 1962).

Encapsulation is the enclosing of one thing within another. It may also be described as the degree to which it is enclosed. System encapsulation encloses system elements and their interactions from the external environment, and usually involves a system boundary that hides the internal from the external; for example, the internal organs of the human body can be optimized to work effectively within tightly defined conditions because they are protected from extremes of environmental change.

Socio-technical systems form what are known as control hierarchies, with systems at a higher level having some ownership of control over those at lower levels. Hitchins (2009) describes how systems form "preferred patterns" which can be used to the enhanced stability of interacting systems hierarchies.

Looking across a hierarchy of systems generally reveals increasing complexity at the higher level, relating to both the structure of the system and how it is used. The term emergence describes behaviors emerging across a complex system hierarchy.

Effectiveness, Adaptation and Learning

Systems effectiveness is a measure of the system's ability to perform the functions necessary to achieve goals or objectives. Ackoff (Ackoff 1971) defines this as the product of the number of combinations of behavior to reach a function and the efficiency of each combination.

Hitchins (2007) describes effectiveness as a combination of **performance** (how well a function is done in ideal conditions), **availability** (how often the function is there when needed), and **survivability** (how likely is it that the system will be able to use the function fully).

System elements and their environment change in a positive, neutral or negative way in individual situations. An adaptive system is one that is able to change itself or its environment if its effectiveness is insufficient to achieve its current or future objectives. Ackoff (Ackoff 1971) defines four types of adaptation, changing the environment or the system in response to internal or external factors.

A system may also **learn**, improving its effectiveness over time, without any change in state or goal.

References

Works Cited

Ackoff, R.L. 1971. "Towards a System of Systems Concepts". *Management Science.* 17(11).

Ackoff, R. 1979. "The Future of Operational Research is Past." *Journal of the Operational Research Society*. 30(2): 93–104, Pergamon Press.

Ashby, W R. 1956. "Chapter 11". *Introduction to Cybernetics*. London, UK: Wiley.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Bertalanffy, L. von. 1975. 'Perspectives on General System Theory. *E. Taschdjian, ed. New York: George Braziller.*

Boardman, J. and B. Sauser. 2008. *Systems Thinking: Coping with 21st Century Problems.* Boca Raton, FL, USA: Taylor & Francis.

Flood, R.L. and E.R. Carson. 1993. *Dealing With Complexity: An Introduction to the Theory and Application of Systems Science*. New York, NY, USA: Plenum Press.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology*. Hoboken, NJ, USA: John Wiley and Sons.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4): 59-63.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Martin J. N. 1997. *Systems Engineering Guidebook.* Boca Raton, FL, USA: CRC Press.

Skyttner, L. 2001. *General Systems Theory: Ideas and Applications.* Singapore: World Scientific Publishing Co. p. 53-69.

Simon, H.A. 1962. "The Architecture of Complexity." *Proceedings of the American Philosophical Society.* 106(6) (Dec. 12, 1962): 467-482.

Simon, H. 1996. *The Sciences of the Artificial*, 3rd ed. Cambridge, MA: MIT Press.

Primary References

Ackoff, R.L. 1971. "Towards a System of Systems Concept." *Management Science.* 17(11).

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4): 59-63.

Additional References

Edson, Robert. 2008. *Systems Thinking. Applied. A Primer*. Arlington, VA, USA: Applied Systems Thinking Institute (ASysT), Analytic Services Inc.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1) (April 2010): 41-43.

Waring, A. 1996. "Chapter 1." *Practical Systems Thinking*. London, UK: International Thomson Business Press.

< Previous Article | Parent Article | Next Article >

Principles of Systems Thinking

This topic forms part of the Systems Thinking knowledge area (KA). It identifies systems principles as part of the basic ideas of systems thinking.

Some additional concepts more directly associated with engineered systems are described, and a summary of system principles associated with the concepts already defined is provided. A number of additional "laws" and heuristics are also discussed.

Systems Principles, Laws, and Heuristics

A principle is a general rule of conduct or behavior (Lawson and Martin 2008). It can also be defined as a basic generalization that is accepted as true and that can be used as a basis for reasoning or conduct (WordWeb 2012c). Thus, systems principles can be used as a basis for reasoning about systems thinking or associated conduct (systems approaches).

Separation of Concerns

A systems approach is focused on a systems-of-interest (SoI) of an open system. This SoI consists of open, interacting subsystems that as a whole interact with and adapt to other systems in an environment. The systems approach also considers the SoI in its environment to be part of a larger, wider, or containing system (Hitchins 2009).

In the What is Systems Thinking? topic, a "systems thinking paradox" is discussed. How is it possible to take a holistic system view while still being able to focus on changing or creating systems?

Separation of concerns describes a balance between considering parts of a system problem or solution while not losing sight of the whole (Greer 2008). Abstraction is the process of taking away characteristics from something in order to reduce it to a set of base characteristics (SearchCIO 2012). In attempting to understand complex situations it is easier to focus on bounded problems, whose solutions still remain agnostic to the greater problem (Erl 2012). This process sounds reductionist, but it can be applied effectively to systems. The key to the success of this approach is ensuring that one of the selected problems is the concerns of the system as a whole. Finding balance between using abstraction to focus on specific concerns while ensuring we continue to consider the whole is at the center of systems approaches. A view is a subset of information observed of one or more entities, such as systems. The physical or conceptual point from which a view is observed is the viewpoint, which can be motivated by one or more observer concerns. Different views of the same target must be both separate, to reflect separation of concerns, and integrated such that all views of a given target are consistent and form a coherent whole (Hybertson 2009). Some sample views of a system are internal (What does it consist of?); external (What are its properties and behavior as a whole?); static (What are its parts or structures?); and dynamic (interactions).

Encapsulation (glossary), which encloses system elements and their interactions from the external environment, is discussed in Concepts of Systems Thinking. Encapsulation is associated with modularity, the degree to which a system's components may be separated and recombined (Griswold 1995). Modularity applies to systems in natural, social, and engineered domains. In engineering, encapsulation is the isolation of a system function within a module and provides precise specifications for the module (IEEE Std. 610.12-1990).

Dualism is a characteristic of systems in which they exhibit seemingly contradictory characteristics that are important for the system (Hybertson 2009). The yin yang concept in Chinese philosophy emphasizes the interaction between dual elements and their harmonization, ensuring a constant dynamic balance through a cyclic dominance of one element and then the other, such as day and night (IEP 2006).

From a systems perspective the interaction, harmonization, and balance between system properties is important. Hybertson (Hybertson 2009) defines **leverage** as the duality between

- **Power**, the extent to which a system solves a specific problem, and
- **Generality**, the extent to which a system solves a whole class of problems.

While some systems or elements may be optimized for one extreme of such dualities, a dynamic balance is needed to be effective in solving complex problems.

Summary of Systems Principles

A set of systems principles is given in Table 1 below. The "Names" segment points to concepts underlying the principle. (See Concepts of Systems Thinking). Following the table, two additional sets of items related to systems principles are noted and briefly discussed: prerequisite laws for design science, and heuristics and pragmatic principles.

Table 1. A Set of Systems Principles. (SEBoK Original)

Name	Statement of Principle
Abstraction	A focus on essential characteristics is important in problem solving because it allows problem solvers to ignore the nonessential, thus simplifying the problem. (Sci-Tech Encyclopedia 2009; SearchCIO 2012; Pearce 2012)
Boundary	A boundary or membrane separates the system from the external world. It serves to concentrate interactions inside the system while allowing exchange with external systems. (Hoagland, Dodson, and Mauck 2001)
Change	Change is necessary for growth and adaptation, and should be accepted and planned for as part of the natural order of things rather than something to be ignored, avoided, or prohibited (Bertalanffy 1968; Hybertson 2009).
Dualism	Recognize dualities and consider how they are, or can be, harmonized in the context of a larger whole (Hybertson 2009)
Encapsulation	Hide internal parts and their interactions from the external environment. (Klerer 1993; IEEE 1990)
Equifinality	In open systems, the same final state may be reached from different initial conditions and in different ways (Bertalanffy 1968). This principle can be exploited, especially in systems of purposeful agents.
Holism	A system should be considered as a single entity, a whole, not just as a set of parts. (Ackoff 1979; Klir 2001)
Interaction	The properties, capabilities, and behavior of a system are derived from its parts, from interactions between those parts, and from interactions with other systems. (Hitchins 2009 p. 60)
Layer Hierarchy	The evolution of complex systems is facilitated by their hierarchical structure (including stable intermediate forms) and the understanding of complex systems is facilitated by their hierarchical description. (Pattee 1973; Bertalanffy 1968; Simon 1996)
Leverage	Achieve maximum leverage (Hybertson 2009). Because of the power versus generality tradeoff, leverage can be achieved by a complete solution (power) for a narrow class of problems, or by a partial solution for a broad class of problems (generality).
Modularity	Unrelated parts of the system should be separated, and related parts of the system should be grouped together. (Griswold 1995; Wikipedia 2012a)
Network	The network is a fundamental topology for systems that forms the basis of togetherness, connection, and dynamic interaction of parts that yield the behavior of complex systems (Lawson 2010; Martin et al. 2004; Sillitto 2010)
Parsimony	One should choose the simplest explanation of a phenomenon, the one that requires the fewest assumptions (Cybernetics 2012). This applies not only to choosing a design, but also to operations and requirements.
Regularity	Systems science should find and capture regularities in systems, because those regularities promote systems understanding and facilitate systems practice. (Bertalanffy 1968)
Relations	A system is characterized by its relations: the interconnections between the elements. Feedback is a type of relation. The set of relations defines the network of the system. (Odum 1994)
Separation of Concerns	A larger problem is more effectively solved when decomposed into a set of smaller problems or concerns. (Erl 2012; Greer 2008)
Similarity/ Difference	Both the similarities and differences in systems should be recognized and accepted for what they are. (Bertalanffy 1975 p. 75; Hybertson 2009). Avoid forcing one size fits all, and avoid treating everything as entirely unique.

Stability/ Change	Things change at different rates, and entities or concepts at the stable end of the spectrum can and should be used to provide a guiding context for rapidly changing entities at the volatile end of the spectrum (Hybertson 2009). The study of complex adaptive systems can give guidance to system behavior and design in changing environments (Holland 1992).
Synthesis	Systems can be created by choosing (conceiving, designing, selecting) the right parts, bringing them together to interact in the right way, and in orchestrating those interactions to create requisite properties of the whole, such that it performs with optimum effectiveness in its operational environment, so solving the problem that prompted its creation" (Hitchins 2008: 120).
View	Multiple views, each based on a system aspect or concern, are essential to understand a complex system or problem situation. One critical view is how concern relates to properties of the whole. (Edson 2008; Hybertson 2009)

The principles are not independent. They have synergies and tradeoffs. Lipson (Lipson 2007), for example, argued that "scalability of open-ended evolutionary processes depends on their ability to exploit functional modularity, structural regularity and hierarchy." He proposed a formal model for examining the properties, dependencies, and tradeoffs among these principles. Edson (Edson 2008) related many of the above principles in a structure called the conceptagon, which he modified from the work of Boardman and Sauser (Boardman and Sauser 2008). Edson also provided guidance on how to apply these principles. Not all principles apply to every system or engineering decision. Judgment, experience, and heuristics (see below) provide understanding into which principles apply in a given situation.

Several principles illustrate the relation of view with the dualism and yin yang principle; for example, holism and separation of concerns. These principles appear to be contradictory but are in fact dual ways of dealing with complexity. Holism deals with complexity by focusing on the whole system, while separation of concerns divides a problem or system into smaller, more manageable elements that focus on particular concerns. They are reconciled by the fact that both views are needed to understand systems and to engineer systems; focusing on only one or the other does not give sufficient understanding or a good overall solution. This dualism is closely related to the systems thinking paradox described in What is Systems Thinking?.

Rosen (Rosen 1979) discussed "false dualisms" of systems paradigms that are considered incompatible but are in fact different aspects or views of reality. In the present context, they are thus reconcilable through yin yang harmonization. Edson (Edson 2008) emphasized viewpoints as an essential principle of systems thinking; specifically, as a way to understand opposing concepts.

Derick Hitchins (Hitchins 2003) produced a systems life cycle theory described by a set of seven principles forming an integrated set. This theory describes the creation, manipulation and demise of engineered systems. These principles consider the factors which contribute to the stability and survival of man made systems in an environment. Stability is associated with the principle of **connected variety**, in which stability is increased by variety plus the **cohesion** and **adaptability** of that variety. Stability is limited by allowable relations, resistance to change, and patterns of interaction. Hitchins describes how interconnected systems tend toward a **cyclic progression**, in which variety is generated, dominance emerges to suppress variety, dominant modes decay and collapse and survivors emerge to generate new variety.

Guidance on how to apply many of these principles to engineered systems is given in the topic Synthesizing Possible Solutions, as well as in System Definition and other knowledge areas in Part 3 of the SEBoK.

Prerequisite Laws of Design Science

John Warfield (Warfield 1994) identified a set of laws of generic design science that are related to systems principles. Three of these laws are stated here:

1. ''Law of Requisite Variety'': A design situation embodies a variety that must be matched by the specifications. The variety includes the diversity of stakeholders. This law is an application of the design science of the Ashby (1956) Law of Requisite Variety, which was defined in the context of cybernetics and states that to successfully regulate a system, the variety of the regulator must be at least as large as the variety of the regulated system.

2. ‘’Law of Requisite Parsimony’’: Information must be organized and presented in a way that prevents human information overload. This law derives from Miller’s findings on the limits of human information processing capacity (Miller 1956). Warfield’s structured dialog method is one possible way to help achieve the requisite parsimony.
3. ‘’Law of Gradation’’: Any conceptual body of knowledge can be graded in stages or varying degrees of complexity and scale, ranging from simplest to most comprehensive, and the degree of knowledge applied to any design situation should match the complexity and scale of the situation. A corollary, called the Law of Diminishing Returns, states that a body of knowledge should be applied to a design situation to the stage at which the point of diminishing returns is reached.

Heuristics and Pragmatic Principles

A heuristic is a common sense rule intended to increase the probability of solving some problem (WordWeb 2012b). In the present context it may be regarded as an informal or pragmatic principle. Maier and Rechtin (Maier and Rechtin 2000) identified an extensive set of heuristics that are related to systems principles. A few of these heuristics are stated here.

- Relationships among the elements are what give systems their added value. This is related to the ‘’Interaction’’ principle.
- Efficiency is inversely proportional to universality. This is related to the ‘’Leverage’’ principle.
- The first line of defense against complexity is simplicity of design. This is related to the ‘’Parsimony’’ principle.
- In order to understand anything, you must not try to understand everything (attributed to Aristotle). This is related to the ‘’Abstraction’’ principle.

An International Council on Systems Engineering (INCOSE) working group (INCOSE 1993) defined a set of “pragmatic principles” for systems engineering (SE). They are essentially best practice heuristics for engineering a system. For example:

- Know the problem, the customer, and the consumer
- Identify and assess alternatives so as to converge on a solution
- Maintain the integrity of the system

Hitchins defines a set of SE principles which include principles of holism and synthesis as discussed above, as well as principles describing how systems problems should be resolved that are of particular relevance to a Systems Approach Applied to Engineered Systems (Hitchins 2009).

References

Works Cited

Ackoff, R. 1979. "The Future of Operational Research is Past." *Journal of the Operational Research Society.* 30(2): 93–104, Pergamon Press.

Ashby, W.R. 1956. "Requisite variety and its implications for the control of complex systems." *Cybernetica.* 1(2):1–17.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. Revised ed. New York, NY: Braziller.

Bertalanffy, L. von. 1975. *Perspectives on General System Theory*. E. Taschdjian, ed. New York: George Braziller.

Boardman, J. and B. Sauser. 2008. *Systems Thinking: Coping with 21st Century Problems*. Boca Raton, FL: Taylor & Francis.

Cybernetics (Web Dictionary of Cybernetics and Systems). 2012. "Principle of Parsimony or Principle of Simplicity." Accessed December 3 2014 at Web Dictionary of Cybernetics and Systems http://pespmc1.vub.ac.

be/ASC/PRINCI_SIMPL.html

Edson, R. 2008. *Systems Thinking. Applied. A Primer*. Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

Erl, T. 2012. "SOA Principles: An Introduction to the Service Orientation Paradigm." Accessed December 3 2014 at Arcitura http://www.soaprinciples.com/p3.php

Greer, D. 2008. "The Art of Separation of Concerns." Accessed December 3 2014 at Aspiring Craftsman http://aspiringcraftsman.com/tag/separation-of-concerns/

Griswold, W. 1995. "Modularity Principle." Accessed December 3 2014 at William Griswold http://cseweb.ucsd.edu/users/wgg/CSE131B/Design/node1.html

Hitchins D. K. 2003. *Advanced systems thinking engineering and management.* Boston, MA, USA: Artech House.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight*. 12(4): 59-63.

Hoagland, M., B. Dodson, and J. Mauck. 2001. *Exploring the Way Life Works*. Burlington, MA, USA: Jones and Bartlett Publishers, Inc.

Holland, J. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA: MIT Press.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL, USA: Auerbach/CRC Press.

IEEE. 1990. *IEEE Standard Glossary of Software Engineering Terminology*. Geneva, Switzerland: Institute of Electrical and Electronics Engineers. IEEE Std 610.12-1990.

IEP (Internet Encyclopedia of Philosophy). 2006. "Yinyang (Yin-yang)." Accessed December 3 2014 at Internet Encyclopedia of Philosophy http://www.iep.utm.edu/yinyang/

INCOSE 1993. *An Identification of Pragmatic Principles - Final Report*. SE Principles Working Group, January 21, 1993.

Klerer, S. “System Management Information Modeling.” *IEEE Communications*. 31(5)May 1993: 38-44.

Klir, G. 2001. *Facets of Systems Science*, 2nd ed. New York, NY: Kluwer Academic/Plenum Publishers.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College, UK.

Lawson, H. and J. Martin. 2008. "On the Use of Concepts and Principles for Improving Systems Engineering Practice." INCOSE International Symposium 2008, 15-19 June 2008, The Netherlands.

Lipson, H. 2007. "Principles of modularity, regularity, and hierarchy for scalable systems." *Journal of Biological Physics and Chemistry.* 7: 125–128.

Maier, M. and E. Rechtin. 2000. *The Art of Systems Architecting, 2nd ed*. Boca Raton, FL: CRC Press.

Miller, G. 1956. "The magical number seven, plus or minus two: some limits on our capacity for processing information." *The Psychological Review.* 63: 81–97.

Odum, H. 1994. *Ecological and General Systems: An Introduction to Systems Ecology (Revised Edition).* Boulder, CO: University Press of Colorado.

Pattee, H. (ed.) 1973. *Hierarchy Theory: The Challenge of Complex Systems*. New York, NY: George Braziller.

Pearce, J. 2012. "The Abstraction Principle." Posting date unknown. Accessed December 3 2014 at Jon Pearce, San Jose State University http://www.cs.sjsu.edu/~pearce/modules/lectures/ood/principles/Abstraction.htm

Rosen, R. 1979. "Old trends and new trends in general systems research." *International Journal of General Systems.* 5(3): 173-184.

Sci-Tech Encyclopedia. 2009. "Abstract Data Type." *McGraw-Hill Concise Encyclopedia of Science and Technology, Sixth Edition*, The McGraw-Hill Companies, Inc.

SearchCIO. 2012. "Abstraction." Accessed December 3 2014 at SearchCIO http://searchcio-midmarket.techtarget.com/definition/abstraction

Sillitto, H. 2010. "Design principles for Ultra-Large-Scale (ULS) Systems." 'Proceedings of INCOSE International Symposium 2010', 12-15 July 2010, Chicago, IL.

Simon, H. 1996. *The Sciences of the Artificial, 3rd ed*. Cambridge, MA: MIT Press.

Warfield, J.N. 1994. *A Science of Generic Design*. Ames, IA: Iowa State University Press.

Wikipedia. 2012a. "Modularity." Accessed December 3 2014 at Wikipedia http://en.wikipedia.org/wiki/Modularity

WordWeb. 2012b. "Dualism." Accessed December 3 2014 at WordWeb http://www.wordwebonline.com/en/DUALISM.

WordWeb. 2012c. "Heuristic." Accessed December 3 2014 at WordWeb http://www.wordwebonline.com/en/HEURISTIC.

WordWeb. 2012d. "Principle." Accessed December 3 2014 at WordWeb http://www.wordwebonline.com/en/PRINCIPLE.

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. Revised ed. New York, NY: Braziller.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Auerbach/CRC Press, Boca Raton, FL.

Klir, G. 2001. *Facets of Systems Science, 2nd ed.* New York: Kluwer Academic/Plenum Publishers.

Additional References

Francois, F. (ed.). 2004. *International Encyclopedia of Systems and Cybernetics,* 2nd ed. Munich, Germany: K. G. Saur Verlag.

Meyers, R. (ed.). 2009. *Encyclopedia of Complexity and Systems Science.* New York, NY: Springer.

Midgley, G. (ed.). 2003. *Systems Thinking.* Thousand Oaks, CA: Sage Publications Ltd.

Volk, T., and J.W. Bloom. (2007). "The use of metapatterns for research into complex systems of teaching, learning, and schooling. Part I: Metapatterns in nature and culture." *Complicity: An International Journal of Complexity and Education.* 4(1): 25—43.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Patterns of Systems Thinking

This topic forms part of the Systems Thinking knowledge area (KA). It identifies systems patterns as part of the basic ideas of systems thinking. The general idea of patterns and a number of examples are described. A brief conclusion discusses the maturity of systems science from the perspective of principles and patterns.

Systems Patterns

This section first discusses definitions, types, and pervasiveness of patterns. Next, samples of basic patterns in the form of hierarchy and network patterns, metapatterns, and systems engineering (SE) patterns are discussed. Then samples of patterns of failure (or "antipatterns") are presented in the form of system archetypes, along with antipatterns in software engineering and other fields. Finally, a brief discussion of patterns as maturity indicators is given.

Pattern Definitions and Types

The most general definition of pattern is that it is an expression of an observed regularity. Patterns exist in both natural and artificial systems and are used in both systems science and systems engineering (SE). Theories in science are patterns. Building architecture styles are patterns. Engineering uses patterns extensively.

Patterns are a representation of similarities in a set or class of problems, solutions, or systems. In addition, some patterns can also represent uniqueness or differences, e.g., uniqueness pattern or unique identifier, such as automobile vehicle identification number (VIN), serial number on a consumer product, human fingerprints, DNA. The pattern is that a unique identifier, common to all instances in a class (such as fingerprint), distinguishes between all instances in that class.

The term pattern has been used primarily in building architecture and urban planning by Alexander (Alexander et al. 1977, Alexander 1979) and in software engineering (e.g., Gamma et al. 1995; Buschmann et al. 1996). Their definitions portray a pattern as capturing design ideas as an archetypal and reusable description. A design pattern provides a generalized solution in the form of templates to a commonly occurring real-world problem within a given context. A design pattern is not a finished design that can be transformed directly into a specific solution. It is a description or template for how to solve a problem that can be used in many different specific situations (Gamma et al. 1995; Wikipedia 2012b). Alexander placed significant emphasis on the pattern role of reconciling and resolving competing forces, which is an important application of the yin yang principle.

Other examples of general patterns in both natural and engineered systems include: conventional designs in engineering handbooks, complex system models such as evolution and predator-prey models that apply to multiple application domains, domain taxonomies, architecture frameworks, standards, templates, architecture styles, reference architectures, product lines, abstract data types, and classes in class hierarchies (Hybertson 2009). Shaw and Garlan (Garlan 1996) used the terms pattern and style interchangeably in discussing software architecture. Lehmann and Belady (Lehmann 1985) examined a set of engineered software systems and tracked their change over time and observed regularities that they captured as evolution laws or patterns.

Patterns have been combined with model-based systems engineering (MBSE) to lead to pattern-based systems engineering (PBSE) (Schindel and Smith 2002, Schindel 2005).

Patterns also exist in systems practice, both science and engineering. At the highest level, Gregory (1966) defined science and design as behavior patterns:

> *The scientific method is a pattern of problem-solving behavior employed in finding out the nature of what exists, whereas the design method is a pattern of behavior employed in inventing things of value which do not yet exist.*

Regularities exist not only as positive solutions to recurring problems, but also as patterns of failure, i.e., as commonly attempted solutions that consistently fail to solve recurring problems. In software engineering these are called antipatterns, originally coined and defined by Koenig (Koenig 1995): An antipattern is just like a pattern, except that instead of a solution it gives something that looks superficially like a solution but isn't one. Koenig's rationale was that if one does not know how to solve a problem, it may nevertheless be useful to know about likely blind alleys. Antipatterns may include patterns of pathologies (i.e., common diseases), common impairment of normal functioning, and basic recurring problematic situations. These antipatterns can be used to help identify the root cause of a problem and eventually lead to solution patterns. The concept was expanded beyond software to include project management, organization, and other antipatterns (Brown et al. 1998; AntiPatterns Catalog 2012).

Patterns are grouped in the remainder of this section into basic foundational patterns and antipatterns (or patterns of failure).

Basic Foundational Patterns

The basic patterns in this section consist of a set of hierarchy and network patterns, followed by a set of metapatterns and SE patterns.

Hierarchy and Network Patterns

The first group of patterns are representative types of hierarchy patterns distinguished by the one-to-many relation type (extended from Hybertson 2009, 90), as shown in the table below. These are presented first because hierarchy patterns infuse many of the other patterns discussed in this section.

Table 1. Hierarchy Patterns. (SEBoK Original)

Relation	Hierarchy Type or Pattern
Basic: Repeating One-to-Many Relation	General: Tree structure
Part of a Whole	Composition (or Aggregation) hierarchy
Part of + Dualism: Each element in the hierarchy is a holon, i.e., is both a whole that has parts and a part of a larger whole	Holarchy (composition hierarchy of holons) (Koestler 1967) - helps recognize similarities across levels in multi-level systems
Part of + Interchangeability: The parts are clonons, i.e., interchangeable	Composition Hierarchy of Clonons (Bloom 2005). Note: This pattern reflects horizontal similarity.
Part of + Self-Similarity: At each level, the shape or structure of the whole is repeated in the parts, i.e., the hierarchy is self-similar at all scales.	Fractal. Note: This pattern reflects vertical similarity.
Part of + Connections or Interactions among Parts	System composition hierarchy
Control of Many by One	Control hierarchy—e.g., a command structure
Subtype or Sub-Class	Type or specialization hierarchy; a type of generalization
Instance of Category	Categorization (object-class; model-metamodel…) hierarchy; a type of generalization

Network patterns are of two flavors. First, traditional patterns are network topology types, such as bus (common backbone), ring, star (central hub), tree, and mesh (multiple routes) (ATIS 2008). Second, the relatively young science of networks has been investigating social and other complex patterns, such as percolation, cascades, power law, scale-free, small worlds, semantic networks, and neural networks (Boccara 2004; Neumann et al. 2006).

Metapatterns

The metapatterns identified and defined in the table below are from (Bloom 2005), (Volk and Bloom 2007), and (Kappraff 1991). They describe a metapattern as convergences exhibited in the similar structures of evolved systems across widely separated scales (Volk and Bloom 2007).

Table 2. Metapatterns. (SEBoK Original)

Name	Brief Definition	Examples
Spheres	Shape of maximum volume, minimum surface, containment	Cell, planet, dome, ecosystem, community
Centers	Key components of system stability	Prototypes, purpose, causation; Deoxyribonucleic acid (DNA), social insect centers, political constitutions and government, attractors
Tubes	Surface transfer, connection, support	Networks, lattices, conduits, relations; leaf veins, highways, chains of command
Binaries Plus	Minimal and thus efficient system	Contrast, duality, reflections, tensions, complementary/symmetrical/reciprocal relationships; two sexes, two-party politics, bifurcating decision process
Clusters, Clustering	Subset of webs, distributed systems of parts with mutual attractions	Bird flocks, ungulate herds, children playing, egalitarian social groups
Webs or Networks	Parts in relationships within systems (can be centered or clustered, using clonons or holons)	Subsystems of cells, organisms, ecosystems, machines, society
Sheets	Transfer surface for matter, energy, or information	Films; fish gills, solar collectors
Borders and Pores	Protection, openings for controlled exchange	Boundaries, containers, partitions, cell membranes, national borders
Layers	Combination of other patterns that builds up order, structure, and stabilization	Levels of scale, parts and wholes, packing, proportions, tiling
Similarity	Figures of the same shape but different sizes	Similar triangles, infant-adult
Emergence	General phenomenon when a new type of functionality derives from binaries or webs.	Creation (birth), life from molecules, cognition from neurons
Holarchies	Levels of webs, in which successive systems are parts of larger systems	Biological nesting from biomolecules to ecosystems, human social nesting, engineering designs, computer software
Holons	Parts of systems as functionally unique	Heart-lungs-liver (holons) of body
Clonons	Parts of systems as interchangeable	Skin cells (clonons) of the skin; bricks in constructing a house
Arrows	Stability or gradient-like change over time	Stages, sequence, orientation, stress, growth, meanders, biological homeostasis, growth, self-maintaining social structures
Cycles	Recurrent patterns in systems over time	Alternating repetition, vortex, spiral, turbulence, helices, rotations; protein degradation and synthesis, life cycles, power cycles of electricity generating plants, feedback cycles
Breaks	Relatively sudden changes in system behavior	Transformation, change, branching, explosion, cracking, translations; cell division, insect metamorphosis, coming-of-age ceremonies, political elections, bifurcation points
Triggers	Initiating agents of breaks, both internal and external	Sperm entering egg or precipitating events of war
Gradients	Continuum of variation between binary poles	Chemical waves in cell development, human quantitative and qualitative values

Systems Engineering Patterns

Some work has been done on various aspects of explicitly applying patterns to SE. A review article of much of this work was written by Bagnulo and Addison (Bagnulo and Addison 2010), covering patterns in general, capability engineering, pattern languages, pattern modeling, and other SE-related pattern topics. Cloutier (Cloutier 2005) discussed applying patterns to SE, based on architecture and software design patterns. Haskins (Haskins 2005), and Simpson and Simpson (Simpson and Simpson 2006) discussed the use of SE pattern languages to enhance the adoption and use of SE patterns. Simpsons identified three high-level, global patterns that can be used as a means of organizing systems patterns:

- Anything can be described as a system.
- The problem system is always separate from the solution system.
- Three systems, at a minimum, are always involved in any system activity: the environmental system, the product system, and the process system.

Haskins (Haskins 2008) also proposed the use of patterns as a way to facilitate the extension of SE from traditional technological systems to address social and socio-technical systems. Some patterns have been applied and identified in this extended arena, described as patterns of success by Rebovich and DeRosa (Rebovich and DeRosa2012). Stevens (Stevens 2010) also discussed patterns in the engineering of large-scale, complex "mega-systems."

A common SE activity in which patterns are applied is in system design, especially in defining one or more solution options for a system-of-interest. See Synthesizing Possible Solutions for a discussion. The more specific topic of using patterns (and antipatterns, as described below) to understand and exploit emergence is discussed in the Emergence topic.

Patterns of Failure: Antipatterns

System Archetypes

The system dynamics community has developed a collection of what are called system archetypes. The concept was originated by Forrester (Forrester 1969), while Senge (Senge 1990) appears to have introduced the system archetype term. According to Braun (2002), the archetypes describe common patterns of behavior that help answer the question, "Why do we keep seeing the same problems recur over time?" They focus on behavior in organizations and other complex social systems that are repeatedly but unsuccessfully used to solve recurring problems. This is why they are grouped here under antipatterns, even though the system dynamics community does not refer to the archetypes as antipatterns. The table below summarizes the archetypes. There is not a fixed set, or even fixed names for a given archetype. The table shows alternative names for some archetypes.

Table 3. System Archetypes. (SEBoK Original)

Name (Alternates)	Description	Reference**
Counterintuitive Behavior	Forrester identified three "especially dangerous" counter-intuitive behaviors of social systems, which correspond respectively to three of the archetypes discussed below: (1) Low-Leverage Policies: Ineffective Actions; (2) High Leverage Policies: Often Wrongly Applied; and (3) Long-Term vs. Short-Term Trade-offs	F1, F2
Low-Leverage Policies: Ineffective Actions (Policy Resistance)	Most intuitive policy changes in a complex system have very little leverage to create change; this is because the change causes reactions in other parts of the system that counteract the new policy.	F1, F3, M
High Leverage Policies: Often Wrongly Applied (High Leverage, Wrong Direction)	A system problem is often correctable with a small change, but this high-leverage solution is typically counter-intuitive in two ways: (1) the leverage point is difficult to find because it is usually far removed in time and place from where the problem appears, and (2) if the leverage point is identified, the change is typically made in the wrong direction, thereby intensifying the problem.	F1, F3, M

Long-Term vs. Short-Term Trade-offs (Fixes that Fail, Shifting the Burden, Addiction)	Short-term solutions are intuitive, but in complex systems there is nearly always a conflict or tradeoff between short-term and long-term goals. Thus, a quick fix produces immediate positive results, but its unforeseen and unintended long-term consequences worsen the problem. Furthermore, a repeated quick fix approach makes it harder to change to a more fundamental solution approach later.	F1, F3, M, S, B
Drift to Low Performance (Eroding Goals, Collapse of Goals)	There is a strong tendency for complex system goals to drift downward. A gap between current state and goal state creates pressure to lower the goal rather than taking difficult corrective action to reach the goal. Over time the continually lowered goals lead to crisis and possible collapse of the system.	F1, F3, M, B
Official Addiction – Shifting the Burden to the Intervener	The ability of a system to maintain itself deteriorates when an intervener provides help and the system then becomes dependent on the intervener	M, S
Limits to Growth (a.k.a. Limits to Success)	A reinforcing process of accelerating growth (or expansion) will encounter a balancing process as the limit of that system is approached and continuing efforts will produce diminishing returns as one approaches the limits.	S, B
Balancing Process with Delay	Delay in the response of a system to corrective action causes the correcting agent to either over-correct or to give up due to no visible progress.	S
Escalation	Two systems compete for superiority, with each escalating its competitive actions to get ahead, to the point that both systems are harmed.	B
Success to the Successful	Growth leads to decline elsewhere. When two equally capable systems compete for a limited resource, if one system receives more resources, it is more likely to be successful, which results in it's receiving even more resources, in a reinforcing loop.	S, B
Tragedy of the Commons	A shared resource is depleted as each system abuses it for individual gain, ultimately hurting all who share it.	H, S, B
Growth and Underinvestment	In a situation where capacity investments can overcome limits, if such investments are not made, then growth stalls, which then rationalizes further underinvestment.	S, B
Accidental Adversaries	Two systems destroy their relationship through escalating retaliations for perceived injuries.	B
Attractiveness Principle	In situations where a system faces multiple limiting or impeding factors, the tendency is to consider each factor separately to select which one to address first, rather than a strategy based on the interdependencies among the factors.	B

**** B—**(Braun 2002); **F1—**(Forrester 1969); **F2—**(Forrester 1995); **F3—**(Forrester 2009); **H—**(Hardin 1968); **M—**(Meadows 1982); **S—**(Senge 1990).

Relations among system archetypes were defined by Goodman and Kleiner (Goodman and Kleiner 1993/1994) and republished in Senge et al. (Senge et al. 1994).

Software and Other Antipatterns

Antipatterns have been identified and collected in the software community in areas that include: Architecture, development, project management, user interface, organization, analysis, software design, programming, methodology, and configuration management (AntiPatterns Catalog 2012, Wikibooks 2012). A brief statement of three of them follows; the first two are organization and the third is software design.

- Escalation of commitment - Failing to revoke a decision when it proves wrong.
- Moral hazard - Insulating a decision-maker from the consequences of his or her decision.
- Big ball of mud - A system with no recognizable structure,

A link between the software community and the system archetypes is represented in a project at the Software Engineering Institute (SEI) (2012), which is exploring the system archetypes in the context of identifying recurring software acquisition problems as "acquisition archetypes." They refer to both types of archetypes as patterns of failure.

Another set of antipatterns in the general systems arena has been compiled by Troncale (Troncale 2010; Troncale 2011) in his systems pathologies project. Sample pathology types or patterns include:

- Cyberpathologies - Systems-level malfunctions in feedback architectures.
- Nexopathologies - Systems-level malfunctions in network architectures or dynamics.
- Heteropathologies - systems-level malfunctions in hierarchical, modular structure & dynamics.

Some treatments of antipatterns, including Senge (Senge 1990) and SEI (SEI 2012), also provide some advice on dealing with or preventing the antipattern.

Patterns and Maturity

Patterns may be used as an indicator of the maturity of a domain of inquiry, such as systems science or systems engineering. In a mature and relatively stable domain, the problems and solutions are generally understood and their similarities are captured in a variety of what are here called patterns. A couple of observations can be made in this regard on the maturity of systems science in support of systems engineering.

In the arenas of physical systems and technical systems, systems science is relatively mature; many system patterns of both natural physical systems and engineered technical systems are reasonably well defined and understood.

In the arena of more complex systems, including social systems, systems science is somewhat less mature. Solution patterns in that arena are more challenging. A pessimistic view of the possibility of science developing solutions to social problems was expressed by Rittel and Webber (Rittel and Webber 1973) in their classic paper on wicked problems: “The search for scientific bases for confronting problems of social policy is bound to fail, because . . . they are ‘wicked’ problems, whereas science has developed to deal with ‘tame’ problems.” A more optimistic stance toward social problems has characterized the system dynamics community. They have been pointing out for over 40 years the problems with conventional solutions to social problems, in the form of the system archetypes and associated feedback loop models. That was an important first step. Nevertheless, they have had difficulty achieving the second step; producing social patterns that can be applied to solve those problems. The antipatterns characterize problems, but the patterns for solving those problems are elusive.

Despite the difficulties, however, social systems do exhibit regularities, and social problems are often solved to some degree. The social sciences and complex systems community have limited sets of patterns, such as common types of organization structures, common macro-economic models, and even patterns of insurgency and counter-insurgency. The challenge for systems science is to capture those regularities and the salient features of those solutions more broadly, and make them explicit and available in the form of mature patterns. Then perhaps social problems can be solved on a more regular basis. As systems engineering expands its scope from the traditional emphasis on technical aspects of systems to the interplay of the social and technical aspects of socio-technical systems, such progress in systems science is becoming even more important to the practice of systems engineering.

References

Works Cited

Alexander, C. 1979. *The Timeless Way of Building*. New York: Oxford University Press.

Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. 1977. *A Pattern Language: Towns – Buildings – Construction*. New York: Oxford University Press.

ATIS. 2008. *ATIS Telecom Glossary 2007*. Washington, D.C.: Alliance for Telecommunications Industry Solutions. Accessed December 3 2014 at ATIS http://www.atis.org/glossary/definition.aspx?id=3516.

Bagnulo, A. and T. Addison. 2010. *State of the Art Report on Patterns in Systems Engineering and Capability Engineering*. Contract Report 2010-012 by CGI Group for Defence R&D Canada – Valcartier. March 2010.

Bloom, J. 2005. "The application of chaos, complexity, and emergent (meta)patterns to research in teacher education." *Proceedings of the 2004 Complexity Science and Educational Research Conference* (pp. 155-191), Sep 30–Oct 3 • Chaffey's Locks, Canada. http://www.complexityandeducation.ca.

Boccara, N. 2004. *Modeling Complex Systems*. New York, NY: Springer-Verlag.

Braun, T. 2002. "The System Archetypes." Accessed December 3 at http:/ / www. albany. edu/ faculty/ gpr/ PAD724/724WebArticles/sys_archetypes.pdf

Brown, W., R. Malveau, H. McCormick, and T. Mowbray. 1998. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley & Sons.

Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. 1996. *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester, U.K.: John Wiley.

Cloutier, R. 2005. "Toward the Application of Patterns to Systems Engineering." 'Proceedings of the Conference on Systems Engineering Research (CSER) 2005, March 23-25, Hoboken, NJ, USA.

Forrester, J. 1969. *Urban Dynamics*. Waltham, MA: Pegasus Communications.

Forrester, J. 1995. "Counterintuitive Behavior of Social Systems." *Technology Review*. 73(3), Jan. 1971: 52-68.

Forrester, J. 2009. Learning through System Dynamics as Preparation for the 21st Century.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.

Goodman, G. and A. Kleiner. 1993/1994. "Using the Archetype Family Tree as a Diagnostic Tool." *The Systems Thinker*. December 1993/January 1994.

Gregory, S. 1966. "Design and the design method," in S. Gregory (ed.). *The Design Method*. London, Engladh: Butterworth.

Hardin, G. 1968. "The Tragedy of the Commons." *Science*. 162(13 December 1968):1243-1248. DOI: 10.1126/science.162.3859.1243.

Haskins, C. 2005. "Application of Patterns and Pattern Languages to Systems Engineering." *Proceedings of the 15th Annual INCOSE International Symposium*. Rochester, NY, July 10-13, 2005.

Haskins, C. 2008. "Using patterns to transition systems engineering from a technological to social context." *Systems Engineering*, 11(2), May 2008: 147-155.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL: Auerbach/CRC Press.

Kappraff, J. (1991). *Connections: The geometric bridge between art and science*. New York, NY: McGraw-Hill.

Koenig, A. 1995. "Patterns and Antipatterns". *Journal of Object-Oriented Programming*. 8(1)March/April 1995: 46–48.

Koestler, A. 1967. *The Ghost in the Machine*. New York, NY: Macmillan.

Lehmann, M. and L. Belady. 1985. *Program Evolution*. London, England: Academic Press.

Meadows, D. 1982. Whole Earth Models and Systems. *The Co-Evolution Quarterly*. Summer 1982: 98-108.

Odum, H.1994. *Ecological and General Systems: An Introduction to Systems Ecology (Revised Edition)*. Boulder, CO: University Press of Colorado.

Rebovich, G. and J. DeRosa 2012. "Patterns of Success in Systems Engineering of IT-Intensive Government Systems." *Procedia Computer Science*. 8(2012): 303 – 308.

Rittel, H. and M. Webber. 1973. "Dilemmas in a general theory of planning." *Policy Sciences*. 4:155–169.

Schindel, W. 2005. "Pattern-based systems engineering: An extension of model-based systems engineering." INCOSE TIES tutorial presented at 2005 INCOSE Symposium, 10-15 July 2005, Rochester, NY.

Schindel, W. and V. Smith. 2002. *Results of applying a families-of-systems approach to systems engineering of product line families.* Technical Report 2002-01-3086. SAE International.

SEI 2012. Patterns of Failure: System Archetypes. Accessed December 3 2014 at SEI http://www.sei.cmu.edu/acquisition/research/pofsa.cfm

Senge, P. 1990. *The Fifth Discipline: Discipline: The Art and Practice of the Learning Organization*. New York, NY: Currency Doubleday.

Senge, P., A. Kleiner, C. Roberts and R. Ross. 1994. *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization*. New York, NY: Currency Doubleday.

Shaw, M. and D. Garlan. 1996. *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ: Prentice Hall.

Simpson, J. and M. Simpson. 2006. "Foundational Systems Engineering Patterns for a SE Pattern Language." *Proceedings of the 16th Annual INCOSE Symposium, July, 2006, Orlando, FL.*

Stevens, R. 2011. *Engineering Mega-Systems: The Challenge of Systems Engineering in the Information Age*. Boca Raton, FL: Auerbach/Taylor & Francis.

Troncale, L. 2010. "Would a Rigorous Knowledge Base in “Systems Pathology” Add to the S.E. Portfolio?" Presented at 2010 LA Mini-Conference, 16 October 2010, Loyola Marymount University, Los Angeles, CA.

Troncale, L. 2011. “Would A Rigorous Knowledge Base in Systems Pathology Add Significantly to the SE Portfolio?" Proceedings of the Conference on Systems Engineering Research (CSER), April 14-16, Redondo Beach, CA.

Volk, T., and J.W. Bloom. 2007. "The use of metapatterns for research into complex systems of teaching, learning, and schooling. Part I: Metapatterns in nature and culture." *Complicity: An International Journal of Complexity and Education*, 4(1): 25—43.

Wikibooks. 2012a. "AntiPatterns." http:/ / en. wikibooks. org/ wiki/ Introduction_to_Software_Engineering/Architecture/Anti-Patterns.

Wikipedia. 2012b. "Software design pattern." http://en.wikipedia.org/wiki/Software_design_pattern

Primary References

Alexander, C. 1979. *The Timeless Way of Building*. New York, NY: Oxford University Press.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. Revised ed. New York, NY: Braziller.

Bloom, J. 2005. "The application of chaos, complexity, and emergent (meta)patterns to research in teacher education." *Proceedings of the 2004 Complexity Science and Educational Research Conference* (pp. 155-191), Sep 30–Oct 3, Chaffey’s Locks, Canada. http://www.complexityandeducation.ca.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Auerbach/CRC Press, Boca Raton, FL.

Additional References

Principia Cybernetica. 1996. "Cybernetics and Systems Theory." Accessed 21 April 2013. Available at: http://pespmc1.vub.ac.be/CYBSYSTH.html

Erl, T. 2009. *SOA: Design Patterns*. Upper Saddle River, NJ: Prentice Hall.

Erl, T. 2008. *SOA: Principles of Service Design*. Upper Saddle River, NJ: Prentice Hall.

Francois, F. (ed.). 2004. *International Encyclopedia of Systems and Cybernetics*, 2nd ed.. Munich, Germany: K. G. Saur Verlag.

Meyers, R. (ed.). 2009. *Encyclopedia of Complexity and Systems Science.* New York, NY: Springer.

Midgley, G. (ed.). 2003. *Systems Thinking.* Thousand Oaks, CA: Sage Publications Ltd.

Principia Cybernetica Web. 2013. "Web Dictionary of Cybernetics and Systems." Accessed 21 April 2013. Available at: http://pespmc1.vub.ac.be/ASC/indexASC.html

---

Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Representing Systems with Models

Representing Systems with Models

A model is a simplified representation of a system at some particular point in time or space intended to promote understanding of the real system. As an abstraction of a system, it offers insight about one or more of the system's aspects, such as its function, structure, properties, performance, behavior, or cost.

Overview

The modeling of systems as holistic, value-providing entities has been gaining recognition as a central process of systems engineering. The use of modeling and simulation during the early stages of the system design of complex systems and architectures can:

- document system functions and requirements,
- assess the mission performance,
- estimate costs,
- evaluate tradeoffs, and
- provide insights to improve performance, reduce risk, and manage costs.

Modeling and analysis can complement testing and evaluation which occur later in the life cycle. In some systems, modeling and simulation may be the only way to fully evaluate performance (e.g., ballistic missile defense) or to evaluate system performance in severe scenarios (e.g., response to weapons of mass destruction attacks on the homeland). Furthermore, advanced simulations, e.g. flight simulators and command and control center simulations, can be a cost-effective technique for personnel training in accompaniment with operational system training (INCOSE 2012).

Modeling serves to make concepts concrete and formal, enhance quality, productivity, documentation, and innovation, as well as to reduce the cost and risk of systems development.

Modeling occurs at many levels: component, subsystem, system, and systems-of-systems; and throughout the life cycle of a system. Different types of models may be needed to represent systems in support of the analysis, specification, design, and verification of systems. This knowledge area provides an overview of models used to represent different aspects of systems.

Modeling is a common practice that is shared by most engineering disciplines, including:

- electrical engineering, which uses electrical circuit design models
- mechanical engineering, which uses three-dimensional computer-aided design models
- software engineering, which uses software design and architecture models.

Each of these disciplines has its own language with its syntax and semantics, serving as a means of communication among professionals in that discipline. Analytic models are used to support power, thermal, structural, and embedded real-time analysis.

Modeling Standards play an important role in defining system modeling concepts that can be represented for a particular domain of interest and enable the integration of different types of models across domains of interest.

Topics

Each part of the Guide to the Systems Engineering Body of Knowledge (SEBoK) is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- What is a Model?
- Why Model?
- Types of Models
- System Modeling Concepts
- Modeling Standards
- Integrating Supporting Aspects into System Models

References

Works Cited

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Primary References

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. Berlin, Germany: Springer Verlag.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies*, rev, B. Seattle, WA: International Council on Systems Engineering. INCOSE-TD-2007-003-02. Accessed April 13, 2015 at http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language,* 2nd Edition. Needham, MA, USA: OMG Press.

Guizzardi, G. 2007. *On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models*. Proceedings of the Databases and Information Systems IV Conference, Amsterdam, Netherlands. Accessed December 4 2014 at ACM http://portal.acm.org/citation.cfm?id=1565425.

INCOSE. 2007. *Systems Engineering Vision 2020*. Seattle, WA, USA: International Council on Systems Engineering. September 2007. INCOSE-TP-2004-004-02.

Wymore, A.W. 1993. *Model-Based Systems Engineering*. Boca Raton, FL, USA: CRC Press, Inc.

Additional References

Holt, J. and S. Perry. 2008. *SysML for systems engineering*. Stevenage: Institution of Engineering and Technology. Accessed December 4 2014 at Ebrary http://site.ebrary.com/id/10263845.

Grobshtein, Y. and D. Dori. 2011. "Generating SysML Views from an OPM Model: Design and Evaluation." *Systems Engineering.* 14(3), Sept.

West, P., J. Kobza, and S. Goerger. 2011. "Chapter 4, Systems Modeling and Analysis," in Parnell, G.S., P.J. Driscoll, and D.L Henderson (eds). *Decision Making for Systems Engineering and Management*, 2nd ed. Wiley Series in Systems Engineering. Hoboken, NJ: Wiley & Sons Inc.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

What is a Model?

This topic provides foundational concepts, such as definitions of a model and a modeling language, and expresses their relationships to modeling tools and model-based systems engineering (MBSE).

Definition of a Model

There are many definitions of the word *model*. The following definitions refer to a model as a representation of selected aspects of a domain of interest to the modeler:

- a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process (DoD 1998);
- a representation of one or more concepts that may be realized in the physical world (Friedenthal, Moore, and Steiner 2009);
- a simplified representation of a system at some particular point in time or space intended to promote understanding of the real system (Bellinger 2004);
- an abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system (Dori 2002); and
- a selective representation of some system whose form and content are chosen based on a specific set of concerns; the model is related to the system by an explicit or implicit mapping (Object Management Group 2010).

In the context of systems engineering, a model that represents a system and its environment is of particular importance to the system engineer who must specify, design, analyze, and verify systems, as well as share information with other stakeholders. A variety of system models are used to represent different types of systems for different modeling purposes. Some of the purposes for modeling systems are summarized in the topic Why Model?, and a simple taxonomy of the different types of models are described in the topic Types of Models. The modeling standards topic refers to some of the standard system modeling languages and other modeling standards that support MBSE.

A model can have different forms as indicated in the first definition above, including a physical, mathematical, or logical representation. A physical model can be a mockup that represents an actual system, such as a model airplane. A mathematical model may represent possible flight trajectories in terms of acceleration, speed, position, and orientation. A logical model may represent logical relationships that describe potential causes of airplane failure, such as how an engine failure can result in a loss of power and cause the airplane to lose altitude, or how the parts of the system are interconnected. It is apparent that many different models may be required to represent a system-of-interest (SoI).

Modeling Language

A physical model is a concrete representation of an actual system that can be felt and touched. Other models may be more abstract representations of a system or entity. These models rely on a modeling language to express their meaning as explained in "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models" (Guizzardi 2007).

Just as engineering drawings express the 3D structure of mechanical and architectural designs, conceptual models are the means by which systems are conceived, architected, designed, and built. The resulting models are the counterparts of the mechanical design blueprint. The difference, however, is that while blueprints are exact representations of physical artifacts with a precise, agreed-upon syntax and long tradition of serving as a means of communication among professionals, conceptual models are just beginning to make headway toward being a complete and unambiguous representation of a system under development. The articles in the special section of Communications of the Association for Computing Machinery (ACM) (Dori 2003) present the abstract world of systems analysis and architecting by means of conceptual modeling, and, how to evaluate, select, and construct models.

Modeling languages are generally intended to be both human-interpretable and computer-interpretable, and are specified in terms of both syntax and semantics.

The abstract syntax specifies the model constructs and the rules for constructing the model. In the case of a natural language like English, the constructs may include types of words such as verbs, nouns, adjectives, and prepositions, and the rules specify how these words can be used together to form proper sentences. The abstract syntax for a mathematical model may specify constructs to define mathematical functions, variables, and their relationship. The abstract syntax for a logical model may also specify constructs to define logical entities and their relationships. A well-formed model abides by the rules of construction, just as a well-formed sentence must conform to the grammatical rules of the natural language.

The concrete syntax specifies the symbols used to express the model constructs. The natural language English can be expressed in text or Morse code. A modeling language may be expressed using graphical symbols and/or text statements. For example, a functional flow model may be expressed using graphical symbols consisting of a combination of graphical nodes and arcs annotated with text; while a simulation modeling language may be expressed using a programming language text syntax such as the C programming language.

The semantics of a language define the meaning of the constructs. For example, an English word does not have explicit meaning until the word is defined. Similarly, a construct that is expressed as a symbol, such as a box or arrow on a flow chart, does not have meaning until it is defined. The language must give meaning to the concept of a verb or noun, and must give specific meaning to a specific word that is a verb or noun. The definition can be established by providing a natural language definition, or by mapping the construct to a formalism whose meaning is defined. As an example, a graphical symbol that expresses $sin(x)$ and $cos(x)$ is defined using a well-defined mathematical formalism for the sine and cosine function. If the position of a pendulum is defined in terms of $sin(\theta)$ and $cos(\theta)$, the meaning of the pendulum position is understood in terms of these formalisms.

Modeling Tools

Models are created by a modeler using modeling tools. For physical models, the modeling tools may include drills, lathes, and hammers. For more abstract models, the modeling tools are typically software programs running on a computer. These programs provide the ability to express modeling constructs using a particular modeling language. A word processor can be viewed as a tool used to build text descriptions using natural language. In a similar way, modeling tools are used to build models using modeling languages. The tool often provides a tool palette to select symbols and a content area to construct the model from the graphical symbols or other concrete syntax. A modeling tool typically checks the model to evaluate whether it conforms to the rules of the language, and enforces such rules

to help the modeler create a well-formed model. This is similar to the way a word processor checks the text to see that it conforms to the grammar rules for the natural language.

Some modeling tools are commercially available products, while others may be created or customized to provide unique modeling solutions. Modeling tools are often used as part of a broader set of engineering tools which constitute the systems development environment. There is increased emphasis on tool support for standard modeling languages that enable models and modeling information to be interchanged among different tools.

Relationship of Model to Model-Based Systems Engineering

The International Council of Systems Engineering (INCOSE) INCOSE Systems Engineering Vision 2020 (2007) defines MBSE as “the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.” In MBSE, the models of the system are primary artifacts of the systems engineering process, and are managed, controlled, and integrated with other parts of the system technical baseline. This contrasts with the traditional document-centric approach to systems engineering, where text-based documentation and specifications are managed and controlled. Leveraging a model-based approach to systems engineering is intended to result in significant improvements in system specification and design quality, lower risk and cost of system development by surfacing issues early in the design process, enhanced productivity through reuse of system artifacts, and improved communications among the system development and implementation teams.

In addition to creating models, the MBSE approach typically includes methods for model management which aim to ensure that models are properly controlled and methods for model validation which aim to ensure that models accurately represent the systems being modeled.

The jointly sponsored INCOSE/Object Management Group (OMG) MBSE Wiki [1] provides additional information on the INCOSE MBSE Initiative including some applications of MBSE and some key topics related to MBSE such as sections on Methodology and Metrics, and Model Management.

The Final Report of the Model Based Engineering (MBE) Subcommittee, which was generated by the the National Defense Industrial Association (NDIA) Modeling and Simulation Committee of the Systems Engineering Division, highlights many of the benefits, risks, and challenges of a model-based approach, and includes many references to case studies of MBE (NDIA 2011).

Brief History of System Modeling Languages and Methods

Many system modeling methods and associated modeling languages have been developed and deployed to support various aspects of system analysis, design, and implementation. Functional modeling languages include the data flow diagram (DFD) (Yourdon and Constantine 1979), Integration Definition for Functional Modeling (IDEF0) (Menzel and Maier 1998), and enhanced functional flow block diagram (eFFBD). Other behavioral modeling techniques include the classical state transition diagram, statecharts (Harel 1987), and process flow diagrams. Structural modeling techniques include data structure diagrams (Jackson 1975), entity relationship diagrams (Chen 1976), and object modeling techniques (Rumbaugh et al. 1991), which combine object diagrams, DFDs, and statecharts.

Some of the recent system modeling methods and languages evolved from these roots and are highlighted in A Survey of Model-Based Systems Engineering (MBSE) Methodologies (Estefan 2008). This survey identifies several candidate MBSE methodologies and modeling languages that can be applied to support an MBSE approach. Additional modeling methods are available from the MBSE Wiki [1] under the section on Methodology and Metrics [2] . The modeling standards section refers to some of the standard system modeling languages and other modeling standards that support MBSE.

References

Works Cited

Bellinger, G. 2004. "Modeling & Simulation: An Introduction," in *Mental Model Musings*. Available at http://www.systems-thinking.org/modsim/modsim.htm.

Chen, P. 1976. "The Entity Relationship Model – Toward a Unifying View of Data." ACM *Transactions on Data Base Systems* 1(1): 9-36.

DoD. 1998. "'DoD Modeling and Simulation (M&S) Glossary" in *DoD Manual 5000.59-M*. Arlington, VA, USA: US Department of Defense. January. P2.13.22. Available at http://www.dtic.mil/whs/directives/corres/pdf/500059m.pdf

Dori, D. 2002. *Object-Process Methodology: A Holistic System Paradigm*. New York, NY, USA: Springer.

Dori, D. 2003. "Conceptual Modeling and System Architecting." *Communications of the ACM*, 46(10), pp. 62-65. [3]

Estefan, J. 2008. "A Survey of Model-Based Systems Engineering (MBSE) Methodologies," rev. B, Seattle, WA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Guizzardi, G. 2007. "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models" Proceedings of Seventh International Baltic Conference. Amsterdam, The Netherlands. Available at http://portal.acm.org/citation.cfm?id=1565425.

Harel, D. 1987. "Statecharts: A Visual Formalism for Complex Systems." *Science of Computer Programming.* 8(3): 231–74.

Jackson, M.A. 1975. *Principles of Program Design.* New York, NY, USA: Academic Press.

Menzel, C. and R.J. Mayer. 1998. "The IDEF Family of Languages." in P. Bernus, K. Mertins, and G. Schmidt (eds.). *Handbook on Architectures for Information Systems.* Berlin, Germany: Springer-Verlag. p. 209-241.

OMG. 2010. *MDA Foundation Model*. Needham, MA, USA: Object Management Group. ORMSC/2010-09-06.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. 1990. *Object-Oriented Modeling and Design.* Upper Saddle River, NJ: Prentice Hall.

Press, Y. and L.L. Constantine. 1976. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Upper Saddle River, NJ: Prentice Hall.

Yourdon E. and Constantine L.L. 1973. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA. 1st Edition.

Primary References

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies,* Rev. B. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02. Available at http://www.incose.org/ProductsPubs/pdf/techdata/MTTC/MBSE_Methodology_Survey_2008-0610_RevB-JAE2.pdf.

Guizzardi, G. 2007. "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models." Proceedings of Seventh International Baltic Conference. Amsterdam, The Netherlands. Available at http://portal.acm.org/citation.cfm?id=1565425.

INCOSE. 2007. *Systems Engineering Vision 2020.* Seattle, WA, USA: International Council on Systems Engineering. September 2007. INCOSE-TP-2004-004-02.

NDIA. 2011. *Final Report of the Model Based Engineering (MBE) Subcommittee*. Arlington, VA, USA: National Defense Industrial Association. Available at: http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Committees/M_S%20Committee/Reports/MBE_Final_Report_Document_(2011-04-22)_Marked_Final_Draft.pdf

Additional References

Downs, E., P. Clare, and I. Coe. 1992. *Structured Systems Analysis and Design Method: Application and Context*. Hertfordshire, UK: Prentice-Hall International.

Eisner, H. 1988. *Computer-Aided Systems Engineering*. Englewood Cliffs, NJ, USA: Prentice Hall.

Harel, D. 1987. "Statecharts: A Visual Formalism for Complex Systems." *Science of Computer Programming.* 8(3): 231–74.

Kossiakoff, A. and W. Sweet. 2003. "Chapter 14" in *Systems Engineering Principles and Practice*. New York, NY, USA: Wiley and Sons.

OMG. "MBSE Wiki." Object Management Group (OMG). Accessed 11 September 2011. Available at: http://www.omgwiki.org/MBSE/doku.php.

Oliver, D., T. Kelliber, and J. Keegan. 1997. *Engineering Complex Systems with Models and Objects.* New York, NY, USA: McGraw-Hill.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.omgwiki.org/MBSE/doku.php
[2] http://www.omgwiki.org/MBSE/doku.php?id=mbse:methodology|
[3] http://esml.iem.technion.ac.il/site/wp-content/uploads/2011/02/article169.pdf

Why Model?

System models can be used for many purposes. This topic highlights some of those purposes, and provides indicators of an effective model, in the context of model-based systems engineering (MBSE).

Purpose of a Model

Models are representations that can aid in defining, analyzing, and communicating a set of concepts. System models are specifically developed to support analysis, specification, design, verification, and validation of a system, as well as to communicate certain information. One of the first principles of modeling is to clearly define the purpose of the model. Some of the purposes that models can serve throughout the system life cycle are

- **Characterizing an existing system:** Many existing systems are poorly documented, and modeling the system can provide a concise way to capture the existing system design. This information can then be used to facilitate maintaining the system or to assess the system with the goal of improving it. This is analogous to creating an architectural model of an old building with overlays for electrical, plumbing, and structure before proceeding to upgrade it to new standards to withstand earthquakes.
- **Mission and system concept formulation and evaluation:** Models can be applied early in the system life cycle to synthesize and evaluate alternative mission and system concepts. This includes clearly and unambiguously defining the system's mission and the value it is expected to deliver to its beneficiaries. Models can be used to explore a trade-space by modeling alternative system designs and assessing the impact of critical system parameters such as weight, speed, accuracy, reliability, and cost on the overall measures of merit. In addition to bounding the system design parameters, models can also be used to validate that the system requirements meet stakeholder needs before proceeding with later life cycle activities such as synthesizing the detailed system design.
- **System design synthesis and requirements flowdown:** Models can be used to support architecting system solutions, as well as flow mission and system requirements down to system components. Different models may be required to address different aspects of the system design and respond to the broad range of system requirements. This may include models that specify functional, interface, performance, and physical requirements, as well as other non-functional requirements such as reliability, maintainability, safety, and security.
- **Support for system integration and verification:** Models can be used to support integration of the hardware and software components into a system, as well as to support verification that the system satisfies its requirements. This often involves integrating lower level hardware and software design models with system-level design models which verify that system requirements are satisfied. System integration and verification may also include replacing selected hardware and design models with actual hardware and software products in order to incrementally verify that system requirements are satisfied. This is referred to as hardware-in-the-loop testing and software-in-the-loop testing. Models can also be used to define the test cases (glossary) and other aspects of the test program to assist in test planning and execution.
- **Support for training:** Models can be used to simulate various aspects of the system to help train users to interact with the system. Users may be operators, maintainers, or other stakeholders. Models may be a basis for developing a simulator (glossary) of the system with varying degrees of fidelity to represent user interaction in different usage scenarios.
- **Knowledge capture and system design evolution:** Models can provide an effective means for capturing knowledge about the system and retaining it as part of organizational knowledge. This knowledge, which can be reused and evolved, provides a basis for supporting the evolution of the system, such as changing system requirements in the face of emerging, relevant technologies, new applications, and new customers. Models can also enable the capture of families of products.

Indicators of an Effective Model

When modeling is done well, a model's purposes are clear and well-defined. The value of a model can be assessed in terms of how effectively it supports those purposes. The remainder of this section and the topics Types of Models, System Modeling Concepts, and Modeling Standards describe indicators of an effective model (Friedenthal, Moore, and Steiner 2012).

Model Scope

The model must be scoped to address its intended purpose. In particular, the types of models and associated modeling languages selected must support the specific needs to be met. For example, suppose models are constructed to support an aircraft's development. A system architecture model may describe the interconnection among the airplane parts, a trajectory analysis model may analyze the airplane trajectory, and a fault tree analysis model may assess potential causes of airplane failure.

For each type of model, the appropriate breadth, depth, and fidelity should be determined to address the model's intended purpose. The model breadth reflects the system requirements coverage in terms of the degree to which the model must address the functional, interface, performance, and physical requirements, as well as other non-functional requirements, such as reliability, maintainability, and safety. For an airplane functional model, the model breadth may be required to address some or all of the functional requirements to power up, takeoff, fly, land, power down, and maintain the aircraft's environment.

The model's depth indicates the coverage of system decomposition from the system context down to the system components. For the airplane example, a model's scope may require it to define the system context, ranging from the aircraft, the control tower, and the physical environment, down to the navigation subsystem and its components, such as the inertial measurement unit; and, perhaps down to lower-level parts of the inertial measurement unit.

The model's fidelity indicates the level of detail the model must represent for any given part of the model. For example, a model that specifies the system interfaces may be fairly abstract and represent only the logical information content, such as aircraft status data; or, it may be much more detailed to support higher fidelity information, such as the encoding of a message in terms of bits, bytes, and signal characteristics. Fidelity can also refer to the precision of a computational model, such as the time step required for a simulation.

Indicators of Model Quality

The quality of a model should not be confused with the quality of the design that the model represents. For example, one may have a high-quality, computer-aided design model of a chair that accurately represents the design of the chair, yet the design itself may be flawed such that when one sits in the chair, it falls apart. A high quality model should provide a representation sufficient to assist the design team in assessing the quality of the design and uncovering design issues.

Model quality is often assessed in terms of the adherence of the model to modeling guidelines and the degree to which the model addresses its intended purpose. Typical examples of modeling guidelines include naming conventions, application of appropriate model annotations, proper use of modeling constructs, and applying model reuse considerations. Specific guidelines are different for different types of models. For example, the guidelines for developing a geometric model using a computer-aided design tool may include conventions for defining coordinate systems, dimensioning, and tolerances.

Model-based Metrics

Models can provide a wealth of information that can be used for both technical and management metrics to assess the modeling effort, and, in some cases, the overall systems engineering (SE) effort. Different types of models provide different types of information. In general, models provide information that enables one to

- assess progress;
- estimate effort and cost;
- assess technical quality and risk; and
- assess model quality.

Models can capture metrics similar to those captured in a traditional document-based approach to systems engineering, but potentially with more precision given the more accurate nature of models compared to documents. Traditional systems engineering metrics are described in *Metrics Guidebook for Integrated Systems and Product Development* (Wilbur 2005).

A model's progress can be assessed in terms of the completeness of the modeling effort relative to the defined scope of the model. Models may also be used to assess progress in terms of the extent to which the requirements have been satisfied by the design or verified through testing. When augmented with productivity metrics, the model can be used to estimate the cost of performing the required systems engineering effort to deliver the system.

Models can be used to identify critical system parameters and assess technical risks in terms of any uncertainty that lies in those parameters. The models can also be used to provide additional metrics that are associated with its purpose. For example, when the model's purpose is to support mission and system concept formulation and evaluation, then a key metric may be the number of alternative concepts that are explored over a specified period of time.

References

Works Cited

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Wilbur, A., G. Towers, T. Sherman, D. Yasukawa, and S. Shreve. 2005. *Metrics Guidebook for Integrated Systems and Product Development*. Seattle, WA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-1995-002-01.

Primary References

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Wilbur, A., G. Towers, T. Sherman, D. Yasukawa, and S. Shreve. 2005. *Metrics Guidebook for Integrated Systems and Product Development*. Seattle, WA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-1995-002-01. Accessed April 13 at https://www.incose.org/ProductsPublications/techpublications/GuideMetrics

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Types of Models

There are many different types of models (glossary) expressed in a diverse array of modeling languages and tool sets. This article offers a taxonomy of model types and highlights how different models must work together to support broader engineering efforts.

Model Classification

There are many different types of models and associated modeling languages to address different aspects of a system and different types of systems. Since different models serve different purposes, a classification of models can be useful for selecting the right type of model for the intended purpose and scope.

Formal versus Informal Models

Since a system model is a representation of a system, many different expressions that vary in degrees of formalism could be considered models. In particular, one could draw a picture of a system and consider it a model. Similarly, one could write a description of a system in text, and refer to that as a model. Both examples are representations of a system. However, unless there is some agreement on the meaning of the terms, there is a potential lack of precision and the possibility of ambiguity in the representation.

The primary focus of system modeling is to use models supported by a well-defined modeling language. While less formal representations can be useful, a model must meet certain expectations for it to be considered within the scope of model-based systems engineering (MBSE). In particular, the initial classification distinguishes between informal and formal models as supported by a modeling language with a defined syntax and the semantics for the relevant domain of interest.

Physical Models versus Abstract Models

The United States “Department of Defense Modeling and Simulation (M&S) Glossary” asserts that “a model can be [a] physical, mathematical, or otherwise logical representation of a system” (1998). This definition provides a starting point for a high level model classification. A physical model is a concrete representation that is distinguished from the mathematical and logical models, both of which are more abstract representations of the system. The abstract model can be further classified as descriptive (similar to logical) or analytical (similar to mathematical). Some example models are shown in Figure 1.

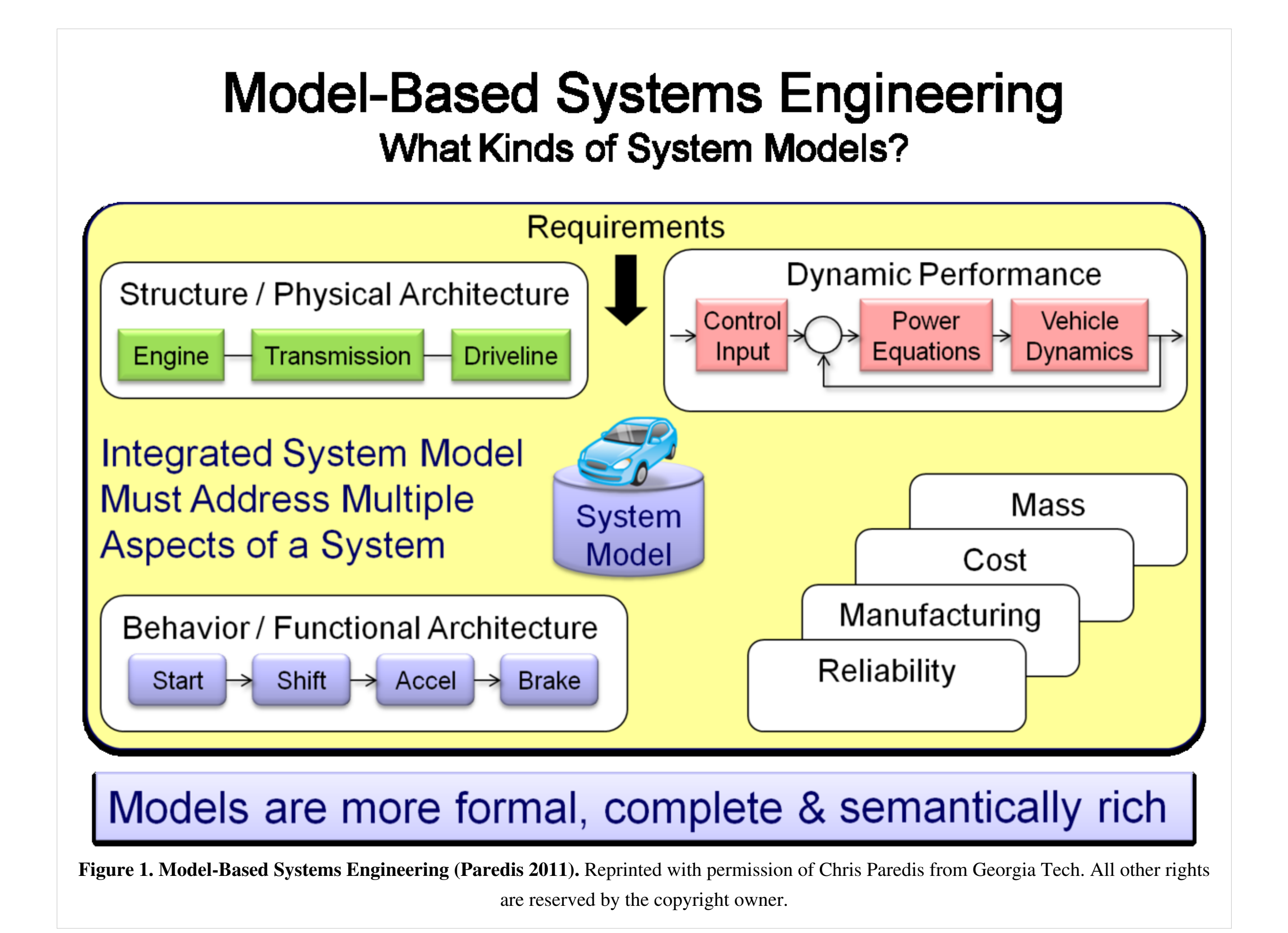


Figure 1. Model-Based Systems Engineering (Paredis 2011). Reprinted with permission of Chris Paredis from Georgia Tech. All other rights are reserved by the copyright owner.

Descriptive Models

A descriptive model describes logical relationships, such as the system's whole-part relationship that defines its parts tree, the interconnection between its parts, the functions that its components perform, or the test cases that are used to verify the system requirements. Typical descriptive models may include those that describe the functional or physical architecture of a system, or the three dimensional geometric representation of a system.

Analytical Models

An analytical model (glossary) describes mathematical relationships, such as differential equations that support quantifiable analysis about the system parameters. Analytical models can be further classified into dynamic and static models. Dynamic models describe the time-varying state of a system, whereas static models perform computations that do not represent the time-varying state of a system. A dynamic model may represent the performance of a system, such as the aircraft position, velocity, acceleration, and fuel consumption over time. A static model may represent the mass properties estimate or reliability prediction of a system or component.

Hybrid Descriptive and Analytical Models

A particular model may include descriptive and analytical aspects as described above, but models may favor one aspect or the other. The logical relationships of a descriptive model can also be analyzed, and inferences can be made to reason about the system. Nevertheless, logical analysis provides different insights than a quantitative analysis of system parameters.

Domain-specific Models

Both descriptive and analytical models can be further classified according to the domain that they represent. The following classifications are partially derived from the presentation on *OWL, Ontologies and SysML Profiles: Knowledge Representation and Modeling* (Web Ontology Language (OWL) & Systems Modeling Language (SysML)) (Jenkins 2010):

- properties of the system, such as performance, reliability, mass properties, power, structural, or thermal models;
- design and technology implementations, such as electrical, mechanical, and software design models;
- subsystems and products, such as communications, fault management, or power distribution models; and
- system applications, such as information systems, automotive systems, aerospace systems, or medical device models.

The model classification, terminology and approach is often adapted to a particular application domain. For example, when modeling organization or business, the behavioral model may be referred to as workflow or process model, and the performance modeling may refer to the cost and schedule performance associated with the organization or business process.

A single model may include multiple domain categories from the above list. For example, a reliability, thermal, and/or power model may be defined for an electrical design of a communications subsystem for an aerospace system, such as an aircraft or satellite.

System Models

System models can be hybrid models that are both descriptive and analytical. They often span several modeling domains that must be integrated to ensure a consistent and cohesive system representation. As such, the system model must provide both general-purpose system constructs and domain-specific constructs that are shared across modeling domains. A system model may comprise multiple views to support planning, requirements, design, analysis, and verification.

Wayne Wymore is credited with one of the early efforts to formally define a system model using a mathematical framework in *A Mathematical Theory of Systems Engineering: The Elements* (Wymore 1967). Wymore established a rigorous mathematical framework for designing systems in a model-based context. A summary of his work can be found in A Survey of Model-Based Systems Engineering (MBSE) Methodologies.

Simulation versus Model

The term simulation, or more specifically computer simulation, refers to a method for implementing a model over time (DoD 1998). The computer simulation includes the analytical model which is represented in executable code, the input conditions and other input data, and the computing infrastructure. The computing infrastructure includes the computational engine needed to execute the model, as well as input and output devices. The great variety of approaches to computer simulation is apparent from the choices that the designer of computer simulation must make, which include

- stochastic or deterministic;
- steady-state or dynamic;
- continuous or discrete; and

- local or distributed.

Other classifications of a simulation may depend on the type of model that is being simulated. One example is an agent-based simulation that simulates the interaction among autonomous agents to predict complex emergent behavior (Barry 2009). They are many other types of models that could be used to further classify simulations. In general, simulations provide a means for analyzing complex dynamic behavior of systems, software, hardware, people, and physical phenomena.

Simulations are often integrated with the actual hardware, software, and operators of the system to evaluate how actual components and users of the system perform in a simulated environment. Within the United States defense community, it is common to refer to simulations as live, virtual, or constructive, where live simulation refers to live operators operating real systems, virtual simulation refers to live operators operating simulated systems, and constructive simulations refers to simulated operators operating with simulated systems. The virtual and constructive simulations may also include actual system hardware and software in the loop as well as stimulus from a real systems environment.

In addition to representing the system and its environment, the simulation must provide efficient computational methods for solving the equations. Simulations may be required to operate in real time, particularly if there is an operator in the loop. Other simulations may be required to operate much faster than real time and perform thousands of simulation runs to provide statistically valid simulation results. Several computational and other simulation methods are described in Simulation Modeling and Analysis (Law 2007).

Visualization

Computer simulation results and other analytical results often need to be processed so they can be presented to the users in a meaningful way. Visualization techniques and tools are used to display the results in various visual forms, such as a simple plot of the state of the system versus time to display a parametric relationship. Another example of this occurs when the input and output values from several simulation executions are displayed on a response surface showing the sensitivity of the output to the input. Additional statistical analysis of the results may be performed to provide probability distributions for selected parameter values. Animation is often used to provide a virtual representation of the system and its dynamic behavior. For example, animation can display an aircraft’s three-dimensional position and orientation as a function of time, as well as project the aircraft’s path on the surface of the Earth as represented by detailed terrain maps.

Integration of Models

Many different types of models may be developed as artifacts of a MBSE effort. Many other domain-specific models are created for component design and analysis. The different descriptive and analytical models must be integrated in order to fully realize the benefits of a model-based approach. The role of MBSE as the models integrate across multiple domains is a primary theme in the International Council on Systems Engineering (INCOSE) INCOSE Systems Engineering Vision 2020 (INCOSE 2007).

As an example, system models can be used to specify the components of the system. The descriptive model of the system architecture may be used to identify and partition the components of the system and define their interconnection or other relationships. Analytical models for performance, physical, and other quality characteristics, such as reliability, may be employed to determine the required values for specific component properties to satisfy the system requirements. An executable system model that represents the interaction of the system components may be used to validate that the component requirements can satisfy the system behavioral requirements. The descriptive, analytical, and executable system model each represent different facets of the same system.

The component designs must satisfy the component requirements that are specified by the system models. As a result, the component design and analysis models must have some level of integration to ensure that the design

model is traceable to the requirements model. The different design disciplines for electrical, mechanical, and software each create their own models representing different facets of the same system. It is evident that the different models must be sufficiently integrated to ensure a cohesive system solution.

To support the integration, the models must establish semantic interoperability to ensure that a construct in one model has the same meaning as a corresponding construct in another model. This information must also be exchanged between modeling tools.

One approach to semantic interoperability is to use model transformations between different models. Transformations are defined which establish correspondence between the concepts in one model and the concepts in another. In addition to establishing correspondence, the tools must have a means to exchange the model data and share the transformation information. There are multiple means for exchanging data between tools, including file exchange, use of application program interfaces (API), and a shared repository.

The use of modeling standards for modeling languages, model transformations, and data exchange is an important enabler of integration across modeling domains.

References

Works Cited

Barry, P.S., M.T.K. Koehler, and B.F. Tivnan. 2009. *Agent-Directed Simulation for Systems Engineering.* McLean, VA: MITRE, March 2009, PR# 09-0267.

DoD. 1998. "'DoD Modeling and Simulation (M&S) Glossary" in *DoD Manual 5000.59-M*. Arlington, VA, USA: US Department of Defense. January 1998.

Wymore, A. 1967. *A Mathematical Theory of Systems Engineering: The Elements*. New York, NY, USA: John Wiley.

Wymore, A. 1993. *Model-Based Systems Engineering*. Boca Raton, FL, USA: CRC Press.

Primary References

Law, A. 2007. *Simulation Modeling and Analysis*, 4th ed. New York, NY, USA: McGraw Hill.

Wymore, A. 1993. *Model-Based Systems Engineering*. Boca Raton, FL, USA: CRC Press.

Additional References

Estefan, J. 2008. *Survey of Candidate Model-Based Systems Engineering (MBSE) Methodologies,* Revision B. Pasadena, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TD-2007-003-02.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL, USA: Auerbach/CRC Press.

INCOSE. 2007. *Systems Engineering Vision 2020.* Seattle, WA, USA: International Council on Systems Engineering. September 2007. INCOSE-TP-2004-004-02.

Rouquette, N. and S. Jenkins. 2010. *OWL Ontologies and SysML Profiles: Knowledge Representation and Modeling.* Proceedings of the NASA-ESA PDE Workshop, June 2010.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Modeling Concepts

A system model represents aspects of a system and its environment. There are many different types of models, as there a variety of purposes for which they are built. It is useful to have a common way to talk about the concepts underlying the many different types of models (e.g., many modeling techniques enable the understanding of system behavior, while others enable the understanding of system structure). This article highlights several concepts used for modeling systems.

Abstraction

Perhaps the most fundamental concept in systems modeling is abstraction, which concerns hiding unimportant details in order to focus on essential characteristics. Systems that are worth modeling have too many details for all of them to reasonably be modeled. Apart from the sheer size and structural complexity that a system may possess, a system may be behaviorally complex as well, with emergent properties, non-deterministic behavior, and other difficult-to-characterize properties. Consequently, models must focus on a few vital characteristics in order to be computationally and intellectually tractable. Modeling techniques address this complexity through various forms of abstraction. For example, a model may assume that structural characteristics of many individual components of a particular type are all the same, ignoring the small order differences between individuals in instances that occur in real life. In that case, those differences are assumed to be unimportant to modeling the structural integrity of those components. Of course, if that assumption is wrong, then the model could lead to false confidence in that structural integrity. There are two key concepts that are applied in regard to modeling different levels of abstraction, which are: view and viewpoint and black-box and white-box modeling, which are described below. Although these two modeling methods are the most widely recognized, different modeling languages and tools employ other techniques as well.

View and Viewpoint

IEEE 1471, a standard for architecture modeling, defines "view" and "viewpoint" as follows:

- View (glossary) - A representation of a whole system from the perspective of a related set of concerns.
- Viewpoint (glossary) - A specification of the conventions necessary for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

Even though IEEE 1471 is focused on architecture models, the concepts of view and viewpoint are general and could apply to models for other purposes as well (IEEE 2000). The viewpoint addresses the concerns of the stakeholders and provides the necessary conventions for constructing a view to address those concerns; therefore, the view represents aspects of the system that address the concerns of the stakeholder. Models can be created to represent the different views of the system. A systems model should be able to represent multiple views of the system to address a range of stakeholder concerns. Standard views may include requirements, functional, structural, and parametric views, as well as a multitude of discipline-specific views to address system reliability, safety, security, and other quality characteristics.

Black-Box and White-Box Models

A very common abstraction technique is to model the system as a black-box, which only exposes the features of the system that are visible from an external observer and hides the internal details of the design. This includes externally visible behavior and other physical characteristics, such as the system’s mass or weight. A white-box model of a system, on the other hand, shows the internal structure and displays the behavior of the system. Black-box and white-box modeling can be applied to the next level of design decomposition in order to create a black-box and

white-box model of each system component.

Conceptual Model

A conceptual model is the set of concepts within a system and the relationships among those concepts (e.g., view and viewpoint). A system conceptual model describes, using one diagram type (such as in Object-Process Methodology (OPM)) or several diagram types (such as in Systems Modeling Language (SysML)) the various aspects of the system. The conceptual model might include its requirements, behavior, structure, and properties. In addition, a system conceptual model is accompanied by a set of definitions for each concept. Sometimes, system concept models are defined using an entity relationship diagram, an object-process diagram (OPD), or a Unified Modeling Language (UML) class diagram.

A preliminary conceptual (or concept) model for systems engineering (Systems Engineering Concept Model) was developed in support of the integration efforts directed toward the development of the Object Management Group (OMG) SysML and the International Organization for Standardization (ISO) AP233 *Data Exchange Standard for Systems Engineering* (ISO 2010). The concept model was originally captured in an informal manner; however, the model and associated concepts were rigorously reviewed by a broad representation of the systems engineering community, including members from the International Council on Systems Engineering (INCOSE), AP233, and SysML development teams.

A fragment from the top level systems engineering concept model is included in Figure 1. This model provides concepts for requirements, behavior, structure and properties of the system, as well as other concepts common to systems engineering and project management, such as stakeholder. The concept model is augmented by a well-defined glossary of terms called the semantic dictionary. The concept model and the semantic dictionary contributed greatly to the requirements for the OMG Systems Modeling Language written in the UML for Systems Engineering Request for Proposal.

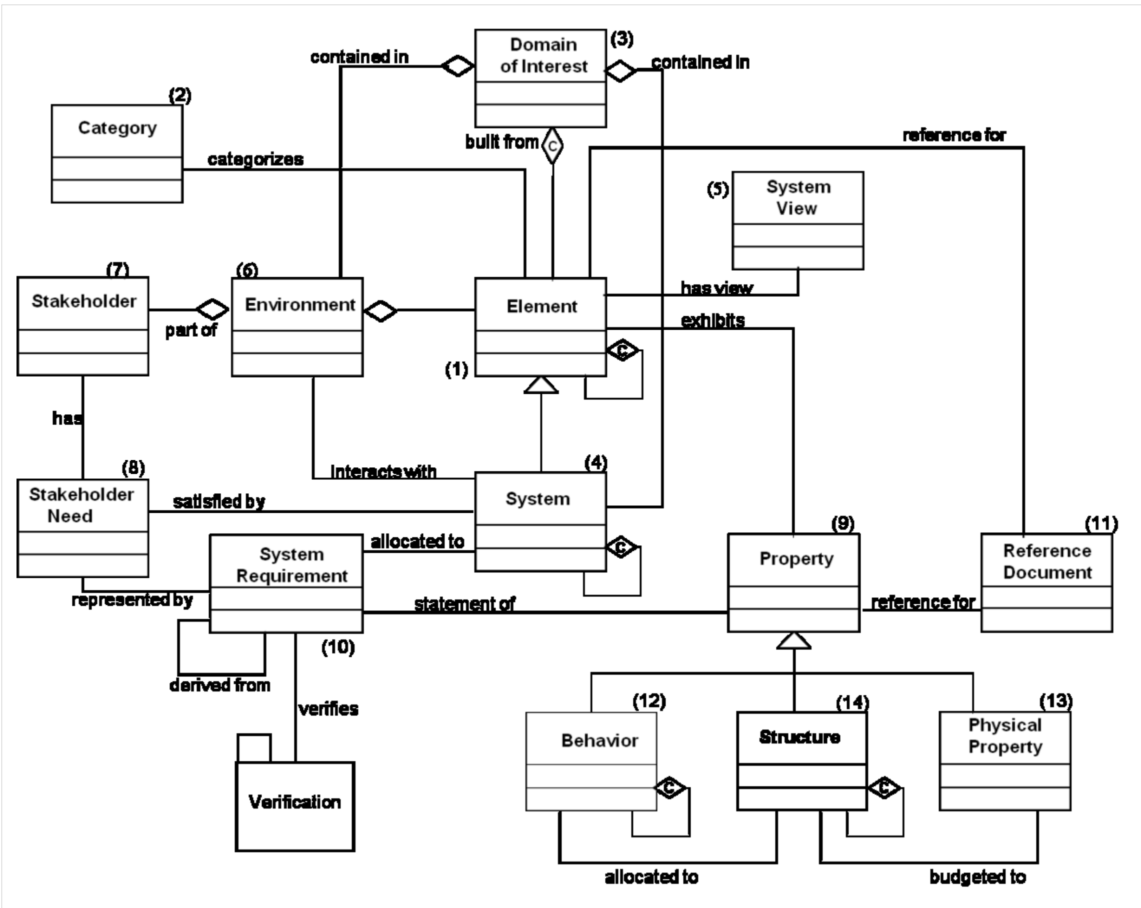


Figure 1. Fragment of the Object Management Group System Concept Model (Oliver 2003, Slide 3). Permission granted by David Oliver on behalf of INCOSE MDSD Working Group. All other rights are reserved by the copyright owner.

A concept model is sometimes referred to as a meta-model, domain meta-model, or schema, and can be used to specify the abstract syntax of a modeling language (refer to the Model Driven Architecture (MDA®) Foundation Model (OMG 2010)). Several other systems engineering concept models have been developed but not standardized. Future standardization efforts should establish a standard systems engineering concept model. The model can then evolve over time as the systems engineering community continues to formalize and advance the practice of systems engineering.

References

Works Cited

IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1471-2000.

ISO. 2010. *OMG System Modeling Language (OMG SysML),* version 1.2. Needham, MA, USA: Object Management Group.

OMG. 2010. *MDA Foundation Model*. Needham, MA, USA: Object Management Group. Document number ORMSC/2010-09-06.

Primary References

ANSI/IEEE. 2000. *Recommended Practice for Architectural Description for Software-Intensive Systems*. New York, NY, USA: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. New York, NY, USA: Springer-Verlag.

Guizzardi, G. 2007. "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models". Proceeding of the 2007 Conference on Databases and Information Systems IV. Available at http://portal.acm.org/citation.cfm?id=1565425.

IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1471-2000.

INCOSE. 2003. *Systems Engineering Concept Model. Draft 12 Baseline.* Seattle, WA: *International Council on Systems Engineering*. Available at http://syseng.omg.org/SE_Conceptual%20Model/SE_Conceptual_Model.htm.

OMG. 2003. *UML for Systems Engineering Request for Proposal*. Needham, MA: Object Management Group. OMG document number ad/2003-3-41. Available at http://www.omg.org/cgi-bin/doc?ad/2003-3-41.

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Integrating Supporting Aspects into System Models

This article discusses the integrated modeling of systems and supporting aspects using Model-Based Systems Engineering methodologies and frameworks. Supporting aspects of systems engineering include:

- Engineering Management
- Project Management
- Requirements Engineering and Management
- Risk Modeling, Analysis, and Management
- Quality Assurance, Testing, Verification, and Validation
- System Integration and Employment
- Analysis of "-ilities" (e.g., Reliability, Availability, Maintainability, Safety, And Security (RAMSS), Manufacturability, Extensibility, Robustness, Resilience, Flexibility, and Evolvability)

These aspects can pertain to physical facets, as well as to functional, structural, behavioral, social, and environmental facets of the core system model. The article focuses on three main aspects:

1. Project and Engineering Management
2. Risk Modeling, Analysis, and Management
3. Requirements Definition and Management

Background

The model-based approach to systems engineering considers the system model as much more than a plain description of the system; the model is the central common basis for capturing, representing, and integrating the various system aspects listed above. The model is essential to the design and understanding of the system as well as to managing its life cycle and evolution. Modeling languages constitute the basis for standardized, formal descriptions of systems, just like natural languages form the basis for human communication.

As systems progressively become more complex and multidisciplinary, the conceptual modeling of systems needs to evolve and will become more critical for understanding complex design (Dori 2002). In addition to facilitating communication among clients, designers, and developers, conceptual modeling languages also assist in clearly describing and documenting various domains, systems, and problems, and define requirements and constraints for the design and development phases (Wand and Weber 2002). The importance of model-based analysis is demonstrated by the variety of conceptual modeling methodologies and frameworks, although *de facto* standards are slow to emerge. While certain disciplines of engineering design, such as structural analysis or circuit design, have established modeling semantics and notation, the conceptual modeling of complex systems and processes has not yet converged on a unified, consolidated modeling framework (Estefan 2007). The challenge is not only to integrate multiple aspects and support the various phases of the system's life cycle, but also to capture the multidisciplinary nature of the system, which has led to the creation of various frameworks. Nevertheless, the information systems analysis paradigm is currently the most widely used, perhaps due to the need to integrate complex systems via information-intensive applications and interactions.

Integrated Modeling of Systems and Projects

This section discusses the integrated modeling of systems and projects and of the project-system relationship (often called Project-Product Integration). The fields of Project Management (glossary) and Systems Engineering (glossary) have been advancing hand-in-hand for the last two decades, due to the understanding that successful projects create successful systems. Many of the main systems engineering resources pay considerable attention to project management and consider it to be a critical process and enabler of systems engineering (INCOSE 2012; NASA 2007; Sage and Rouse 2011). The integration of system-related aspects and concepts into project plans is more common than the integration of project-related aspects and concepts into system models. Because the project is a means to an end, it is the process that is expected to deliver the system. Indeed, project activities are often named after or in accord with the deliverables that they are aimed at facilitating (e.g., "console design", "software development", "hardware acquisition", or "vehicle assembly"). Each is a function name, consisting of an object (noun), or the system to be attained, and a process (verb), being the project or part of the project aimed at attaining the end system.

The specific process associated with each of these examples refers to different stages or phases of the project and to different maturity levels of the system or sub-system it applies to. Moreover, the mere inclusion of system and part names in activity names does not truly associate system model artifacts with these activities. Overall, it is not truly possible to derive the set of activities associated with a particular part or functionality of the system which that will be delivered by the project. Project-Product integration is not straightforward, as project models and system models are traditionally disparate and hardly interface. A model-based approach to project-system integration follows a system-centric paradigm and focuses on incorporating project-related aspects and concepts into the core system model, as opposed to the project-centric approach described in the previous paragraph. Such aspects and concepts include schedule, budget and resources, deliverables, work-packages, constraints and previous relations. The integrated system-project model should provide useful information on the mutual effects of project activities and system components and capabilities. Some examples of integrated system-project modeling include the following:

- The set of project activities associated with a particular system component, feature, or capability.

- The set of resources required for performing a task of designing or developing a particular component of the system.
- The team or subcontractor responsible for delivering each system component.
- The preexisting dependencies between activities of system components deployment.
- The cost associated with each system component, feature, or capability.
- The parts of the system negotiated for each delivery, deployment, build, release, or version.

The Work Breakdown Structure (WBS) is designed to support the division of the project scope (work content) amongst the individuals and organizations participating in the project (Golany and Shtub 2001). The WBS is traditionally organization or activity-oriented; however, one of its main cornerstones focuses on the deliverable, which corresponds to the system, sub-system, component, or a capability or feature of one or more of these. A deliverable-oriented WBS, in which the high-level elements correspond to primary sub-systems, is advocated, as it is likely to allow the WBS to be more product-oriented (Rad 1999). An integrated approach to project planning and system modeling (Sharon and Dori 2009) merges the system model with the project’s WBS using Object-Process Methodology (OPM) (Dori 2002). The unified OPM model captures both the project activities and the system components and functionalities.

The Design Structure Matrix (DSM) is a common method for enhancing and analyzing the design of products and systems. DSMs can be component-based, task-based, parameter-based, or team-based (Browning 2001). A DSM for an OPM-based project-product model derives a hybrid DSM of project activities and system building blocks from the unified OPM model, accounting for dependencies between project activities and system components, as well as replacing the two monolithic and separate component-based and task-based DSM views (Sharon, De-Weck, and Dori 2012). The underlying OPM model assures model consistency and traceability. The integrated project-product OPM model includes both a diagram and an equivalent auto-generated textual description. The DSM derived from this model visualizes a dependency loop comprising both system components and project activities.

Another model-based approach (Demoly et al. 2010) employs System Modeling Language (SysML) in order to create various views that meet the needs of various system stakeholders, such as the project/process manager. The approach includes both product-oriented and process-oriented views.

Integrated Modeling of Systems and Requirements

Requirements are statements that describe operational, functional, or design-related aspects of a system. Requirements definition and management is an important SE process, as it both initiates and facilitates the entire SE effort by defining the expected functions and performance of the engineered system. Several challenges associated with requirements include:

- Defining the requirements in a structured, controlled manner.
- Tracing these requirements to system components, aspects, and decisions.
- Testing and verifying compliance of the system with these requirements.

The extension of conceptual system models to include requirements has several significant benefits:

1. Requirements provide the rationale for the system's architecture and design by making and justifying architectural and design decisions based on specific requirements.
2. Modeling the internal logic and the hierarchy and dependency relations among requirements enables identification and elimination of redundant and contradictory requirements.
3. Responsibility for satisfying specific requirements can often be assigned to teams and persons responsible for delivering various system components. While the advantages of having good requirements engineering is clear, it is often a challenge to directly trace requirements to specific system artifacts, especially when the requirements are defined in a holistic, solution-independent manner.

There are several methods to incorporate requirements into system models, including SysML Requirements Engineering and Object-Process Methodology(OPM)-based Requirements Engineering and Authoring.

SysML-Based Requirements Engineering

The SysML requirements diagram makes it possible to capture the requirements and the relations among them in a visual manner, which is more intuitive than the textual manner in which requirements are traditionally edited and managed. The diagram was added to the basic set of UML diagrams that formed the basis for SysML (Friedenthal, Moore, and Steiner 2006), and is not a native UML diagram. Tracing requirements to the system blocks and artifacts satisfying them can be captured in the SysML Block Definition Diagram, which is primarily designated to capture the relations among types of system elements and components. The < > link between the block and the requirement captures the trace.

OPM-Based Requirements Engineering

Object-Process Methodology (OPM) is a methodology and language for conceptual modeling of complex systems and processes with a bimodal textual and graphical representation (Dori 2002). OPM's textual representation is coordinated with the graphical representation; additionally, each visual model construct in the Object-Process Diagram (OPD) is described by a formal structured textual statement in Object-Process Language (OPL), which is a subset of natural English. OPM facilitates model-based requirements engineering, authoring, and specification, in three possible modes:

1. OPM can be used to generate conceptual models which initially focus on the requirements level—the problem domain, rather than the design level or the solution domain, which facilitates automated model-based requirements generation (Blekhman and Dori 2011). The requirements model is solution-neutral and it can be the basis for one or more architectural solutions for achieving the functions specified in the requirements.
2. Utilizing OPM in order to generate requirement-oriented OPDs in a manner similar to the SysML Requirements Diagram enables an engineer to capture the requirements specification as the skeleton for the system model. User-defined tagged structural relations, such as "is realized by" or is allocated to", provide for associating requirements with system model functions (objects and processes that transform them). This approach is similar to the SysML requirements diagram; however, instead of using a unique notation in a separate diagram type, the requirements are seamlessly incorporated into the single system model.
3. OPM can be used for the purpose of generating visual system models from formally specified requirements by tracing the textually authored requirements to system model inserts and artifacts (Dori et al. 2004).

Integrating Risk into System Models

Risk is an expression and a measure of the negative or adverse impact of uncertainty. Risk exists whenever uncertainty can lead to several results, of which some may be negative (adverse) and some positive. A system faces risks from other systems or from the environment, and it can also pose risks to other systems or to the environment. Systems are characterized by such attributes, such as: goals, objectives, inputs, outputs, variables, parameters, processes, events, states, subsystems, interfaces, mechanisms, and methods. System vulnerability is the system's total potential to be harmed or negatively affected in any one of these attributes. Analogously, system harmfulness is the system's total potential to harm others or to generate negative effects, which can be manifested in one or more of these attributes (Haimes 2009). Model-based risk analysis (MBRA) enables structured analysis and risk-related process control. Several model-based risk analysis approaches are available in the literature. MBRA is presently common in the information technology and information security domains more than the systems engineering domain; however, some of the methods are generally applicable to complex systems as well. The ISO-IEC-IEEE collaborative software development and operation lifecycle standard (ISO and IEC 2004) proposes a concurrent approach to IT Risk Management. This approach consists of six main activities:

- Plan and Implement Risk Management
- Manage the Project Risk Profile
- Perform Risk Analysis
- Perform Risk Monitoring
- Perform Risk Treatment
- Evaluate the Risk Management Process

These activities are executed concurrently, affect and provide feedback to each other, and interact with other software life cycle processes, such as the technical management and the design processes (ISO and IEC 2004).

The CORAS approach (Fredriksen et al. 2002; den Braber et al. 2006; Lund, Solhaug, and Stølen 2011) is a UML-derivative for IT security risk modeling and assessment. This framework consists mostly of the UML use case (UC) diagram, extended for misuse cases. Additional notation was added to the UC notation in order to capture risk sources, effects, and results (e.g., the "bad actor" icon, moneybag for asset-in-risk). A misuse diagram can include, for example, the risk of loss of legal protection of proprietary know-how due to information theft and distribution by an unfaithful employee. The treatment for the risk source of insufficient security policy, which contributes to the above risk, is illustrated in a separate treatment diagram.

A quantitative risk assessment method for component-based systems (Grunske and Joyce 2008) supports component vulnerability analysis and specification using modular attack trees. In addition, it provides attacker profiling, which enables supporting econometric approaches to risk response. The methodology utilizes SysML as its underpinning language, and especially the SysML block definition diagram and parametric diagram, in order to capture parametric relations and constraints as a means to defining risk profiles.

System-Theoretic Accident Model and Processes (STAMP) is a method for system and component design for safety (Leveson 2011). STAMP reformulates the safety problem as a control problem as opposed to a reliability problem. STAMP is optimized for safety-oriented systems engineering and design and for hazard avoidance and mitigation, specifically in complex socio-technical systems. A model-based adaptation of STAMP was also proposed (Leveson 2004) and was implemented in various safety-critical and mission-critical systems, including aircraft collision avoidance systems (CAS) (Leveson 2004) and ballistic missile defense systems (Pereira, Lee, and Howard 2006). Risk-Oriented Systems Engineering (ROSE) (Mordecai and Dori 2013) is a method based on Object-Process Methodology (OPM) for integrating risk into system models. Being system-centric, ROSE is responsible for capturing risk layers and aspects on top of and in sync with the core system model, while improving and immunizing it against captured risks, as well as for generating system robustness and resilience by design in response to various risk-posing scenarios. The risk handling meta-model includes risk mitigation during the design phase and risk response during the operational phase.

References

Works Cited

Blekhman, A. and D. Dori. 2011. "Model-Based Requirements Authoring - Creating Explicit Specifications with OPM." In 6th International Conference on Systems Engineering. Herzeliyya, Israel.

Browning, T.R. 2001. "Applying the Design Structure Matrix to System Decomposition and Integration Problems: a Review and New Directions." IEEE Transactions on Engineering Management 48 (3): 292–306. doi:10.1109/17.946528. Accessed December 4 2014 at IEEE http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=946528.

Demoly, F., D. Monticolo, B. Eynard, L. Rivest, and S. Gomes. 2010. "Multiple Viewpoint Modelling Framework Enabling Integrated Product–process Design." International Journal on Interactive Design and Manufacturing (IJIDeM) 4 (4) (October 12): 269–280. doi:10.1007/s12008-010-0107-3. Accessed December 4 2014 at Springer http://link.springer.com/10.1007/s12008-010-0107-3.

Den Braber, F., G. Brændeland, H.E.I. Dahl, I. Engan, I. Hogganvik, M.S. Lund, B. Solhaug, K. Stølen, and F. Vraalsen. 2006. *The CORAS Model-based Method for Security Risk Analysis*. SINTEF, Oslo. Oslo: SINTEF.

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. New York, NY, USA: Springer-Verlag.

Dori, D., N. Korda, A. Soffer, and S. Cohen. 2004. "SMART: System Model Acquisition from Requirements Text." Lecture Notes in *Computer Science: Business Process Management* 3080: 179–194. Accessed December 4 2014 at Springer http://link.springer.com/chapter/10.1007/978-3-540-25970-1_12.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies,* Rev. B. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Friedenthal, S., A. Moore, and R. Steiner. 2006. "OMG Systems Modeling Language (OMG SysML™) Tutorial" (July).

Golany, B. and A. Shtub. 2001. "Work Breakdown Structure." In Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc. 1263–1280.

Grunske, L. and D. Joyce. 2008. "Quantitative Risk-based Security Prediction for Component-based Systems with Explicitly Modeled Attack Profiles." Journal of Systems and Software 81 (8): 1327–1345. Haimes, YY. 2009. "On the Complex Definition of Risk: A Systems-Based Approach." *Risk Analysis.* 29 (12): 1647–1654. Accessed December 4 2014 at Wiley http://onlinelibrary.wiley.com/doi/10.1111/j.1539-6924.2009.01310.x/full.

ISO/IEC/IEEE. 2004. *Systems and software engineering - Life cycle processes - Risk management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC/IEEE 16085:2006.

Leveson, N.G. 2004. "Model-based Analysis of Socio-technical Risk." Cambridge, MA: Massachusetts Institute of Technology (MIT) Working Paper Series. ESD-WP-2004-08.

Leveson, N.G. 2011. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: MIT Press.

Lund, M.S., B. Solhaug, and K. Stølen. 2011. *Model-Driven Risk Analysis: The CORAS Approach*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-12323-8. Accessed December 4 2014 at Springer http://www.springerlink.com/index/10.1007/978-3-642-12323-8.

Mordecai, Y., and D. Dori. 2013. "Model-Based Risk-Oriented Robust Systems Design with Object-Process Methodology." *International Journal of Strategic Engineering Asset Management*. TBD (CESUN 2012 Special Issue).

NASA. 2007. *NASA Systems Engineering Handbook|Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Pereira, S.J., G. Lee, and J. Howard. 2006. "A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System". Vol. 1606. Accessed December 4 2014 at Defense Technical Information Center http:/ / oai. dtic. mil/ oai/ oai?verb=getRecord& metadataPrefix=html& identifier=ADA466864.

Rad, P.F. 1999. "Advocating a Deliverable-oriented Work Breakdown Structure." Sage, Andrew P., and William B. Rouse. 2011. *Handbook of Systems Engineering and Management*. Edited by A.P. Sage and W.B. Rouse. 2nd ed. John Wiley & Sons.

Sharon, A., O.L. de-Weck, and D. Dori. 2012. "Improving Project-Product Lifecycle Management with Model-Based Design Structure Matrix : A Joint Project Management and Systems Engineering Approach." *Systems Engineering*: 1–14. doi:10.1002/sys.

Sharon, A., and D. Dori. 2009. "A Model-Based Approach for Planning Work Breakdown Structures of Complex Systems Projects." In Proc. 14th IFAC Symposium on Information Control Problems in Manufacturing.

Wand, Y., and R. Weber. 2002. "Research Commentary: Information Systems and Conceptual Modeling?A Research Agenda." *Information Systems Research.* 13(4) (December): 363–376. doi:10.1287/isre.13.4.363.69.

Primary References

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. New York, NY, USA: Springer-Verlag.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies,* Rev. B. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Golany, B. and A. Shtub. 2001. "Work Breakdown Structure." In Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc. 1263–1280.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

Kristiansen, B.G. and K. Stolen. 2002. "The CORAS Framework for a Model-based Risk Management Process." In Lecture Notes In, edited by Stuart Anderson, Massimo Felici, and SandroEditors Bologna, 2434:94–105. Springer-Verlag. doi:10.1007/3-540-45732-1_11.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Modeling Standards

Different types of models are needed to support the analysis, specification, design, and verification of systems. The evolution of modeling standards enables the broad adoption of Model-Based Systems Engineering (MBSE).

Motivation for Modeling Standards

Modeling standards play an important role in defining agreed-upon system modeling concepts that can be represented for a particular domain of interest and enable the integration of different types of models across domains of interest. Modeling standards are extremely important to support MBSE, which aims to integrate various system aspects across various disciplines, products, and technologies.

Standards for system modeling languages can enable cross-discipline, cross-project, and cross- organization communication. This communication offers the potential to reduce the training requirements for practitioners who only need to learn about a particular system and enables the reuse of system artifacts. Standard modeling languages also provide a common foundation for advancing the practice of systems engineering, as do other systems engineering standards.

Types of Modeling Standards

Many different standards apply to systems modeling. Modeling standards include standards for modeling languages, data exchange between models, and the transformation of one model to another to achieve semantic interoperability. Each type of model can be used to represent different aspects of a system, such as representing the set of system components and their interconnections and interfaces, or to represent a system to support performance analysis or reliability analysis.

The following is a partial list of representative modeling standards, which also includes the common acronym, when applicable, and a reference as to where additional information can be found on the topic.

Modeling Languages for Systems

Descriptive Models - These standards apply to general descriptive modeling of systems:

- Functional Flow Block Diagram (FFBD) (Oliver, Kelliher, and Keegan 1997)
- Integration Definition for Functional Modeling (IDEF0) (NIST 1993)
- Object-Process Methodology (OPM) [[1]] [[2]] (Dori 2002; ISO 19450 PAS - Publicly Available Specification in progress)
- Systems Modeling Language (SysML)(OMG 2010a)
- Unified Profile for United States Department of Defense Architecture Framework (DoDAF) and United Kingdom Ministry of Defense Architecture Framework (MODAF) (OMG 2011e)
- Web ontology language (OWL) (W3C 2004b)

Analytical Models and Simulations - These standards apply to analytical models and simulations:

- Distributed Interactive Simulation (DIS) (IEEE 1998)
- High-Level Architecture (HLA) (IEEE 2010)
- Modelica (Modelica Association 2010)
- Semantics of a Foundational Subset for Executable Unified Modeling Language (UML) Models (FUML) (OMG 2011d)

Data Exchange Standards

These standards enable the exchange of information between models:

- Application Protocol for Systems Engineering Data Exchange (ISO 10303-233) (AP-233) (ISO 2005)
- Requirements Interchange Format (ReqIF) (OMG 2011c)
- Extensible Mark-Up Language -(XML) Metadata Interchange (XMI) (OMG 2003a)
- Resource Description Framework (RDF) (W3C 2004a)

Model Transformations

These standards apply to transforming one model to another to support semantic interoperability:

- Query View Transformations (QVT) (OMG 2011b)
- Systems Modeling Language (SysML)-Modelica Transformation (OMG 2010c)
- OPM-to-SysML Transformation (Grobshtein and Dori 2011)

General Modeling Standards

These standards provide general frameworks for modeling:

- Model-driven architecture (MDA®) (OMG 2003b)
- IEEE 1471-2000 - Recommended Practice for Architectural Description of Software-Intensive Systems (ANSI/IEEE 2000) (ISO/IEC 2007)

Other Domain-Specific Modeling Standards

Software Design Models

These standards apply to modeling application software and/or embedded software design:

- Architecture Analysis and Design Language (AADL) (SAE 2009)
- Modeling and Analysis for Real-Time and Embedded Systems (MARTE) (OMG 2009)
- Unified Modeling Language (UML) (OMG 2010b)

Hardware Design Models

These standards apply to modeling hardware design:

- Very-High-Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) (IEEE 2008)

Business Process Models

These standards apply to modeling business processes:

- Business Process Modeling Notation (BPMN) (OMG 2011a)

References

Works Cited

ANSI/IEEE. 2000. *Recommended Practice for Architectural Description for Software-Intensive Systems*. New York, NY: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

Grobshtein, Y. and D. Dori. 2011. "Generating SysML Views from an OPM Model: Design and Evaluation." *Systems Engineering*, 14 (3), Sept. 2011.

IEEE. 1998. *Distributed Interactive Simulation (DIS)*. Washington, DC: Institute for Electrical and Electronic Engineers. IEEE 1278.1-1995. Accessed December 4 2014 at IEEE http://standards.ieee.org/develop/project/1278.2.html.

IEEE. 2008. *VHSIC hardware description language (VHDL)*. Washington, DC: Institute of Electrical and Electronics Engineers. IEEE Standard 1076-2008. Accessed December 4 2014 at IEEE http://standards.ieee.org/findstds/standard/1076-2008.html.

IEEE. 2010. *Standard for High Level Architecture*. Washington, DC: Institute for Electrical and Electronic Engineers. IEEE Standard 1516. Accessed December 4 2014 at IEEE http://standards.ieee.org/develop/intl/intlstds.html

ISO. 2005. *Application Protocol for Systems Engineering Data Exchange*. Geneva, Switzerland: International Organization for Standardization. ISO 10303-233. Accessed December 4 2014 at ISO http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=55257.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering — Architecture Description.* Geneva, Switzerland: International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers. December 1, 2011. ISO/IEC/IEEE 42010:2011. Accessed December 4 2014 at ISO http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=50508.

Modelica Association. 2010. *Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification, Version 3.2*. Modelica Association. Accessed December 4 2014 at Modelica https://www.modelica.org/documents/ModelicaSpec32.pdf.

NIST. 1993. *Integration Definition for Functional Modeling (IDEF0)*. Gaithersburg, MD: National Institute for Standards and Technologies. Accessed December 4 2014 at IDEF http://www.idef.com/IDEF0.htm.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering Complex Systems with Models and Objects*. New York, NY, USA: McGraw Hill.

OMG 2003a. *XML Metadata Interchange (XMI),* Version 1.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/XML/.

OMG. 2003b. *Model driven architecture (MDA®),* Version 1.0.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/mda.

OMG. 2009. *Modeling and Analysis for Real-Time and Embedded Systems (MARTE),* Version 1.0. Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/MARTE/1.0/.

OMG. 2010a. *OMG Systems Modeling Language (SysML),* Version 1.2. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at SysML forum http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf.

OMG. 2010b. *Unified Modeling Language™ (UML),* Version 2.. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/UML/.

OMG. 2010c. *SysML-Modelica Transformation Specification*, Beta Version. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/SyM/.

OMG. 2011a. *Business Process Modeling Notation (BPMN),* Version 2.0. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/BPMN/2.0/

OMG. 2011b. *Query View Transformations (QVT),* Version 1.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/QVT/1.1/.

OMG. 2011c. *Requirements Interchange Format (ReqIF),* Version 1.0.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/ReqIF/.

OMG. 2011d. *Semantics of a Foundational Subset for Executable UML Models (FUML),* Version 1.0. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/FUML/1.0/.

OMG. 2011e. *Unified Profile for DoDAF and MODAF (UPDM),* Version 1.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/UPDM/.

SAE. 2009. *Architecture Analysis & Design Language (AADL)*. Warrendale, PA, USA: SAE International. Accessed December 4 2014 at Society of Automotive Engineers http://standards.sae.org/as5506a/.

W3C. 2004a. *Resource Description Framework (RDF),* Version 1.0. World Wide Web Consortium. Accessed December 4 2014 at World Wide Web Consortium http://www.w3.org/RDF/.

W3C. 2004b. *Web ontology language. (OWL)*. World Wide Web Consortium. Accessed December 4 2014 at World Wide Web Consortium http://www.w3.org/2004/OWL.

Primary References

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm.* Berlin, Germany: Heidelberg; New York, NY, USA: Springer Verlag.

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Additional References

Fritzon, P. 2004. *Object-oriented modeling and simulation with Modelica 2.1*. New York, NY, USA: Wiley Interscience and IEEE Press.

Bibliowicz, A. and D. Dori. *A Graph Grammar-Based Formal Validation of Object-Process Diagrams*. Software and Systems Modeling, 11, (2) pp. 287-302, 2012.

Blekhman, A. and D. Dori. "Model-Based Requirements Authoring." INCOSE 2011 – the 6th International conference on System Engineering. March, 2011.

Dori, D., R. Feldman, and A. Sturm. *From conceptual models to schemata: An object-process-based data warehouse construction method."* Information Systems. *33: 567–593, 2008.*

Osorio, C.A., D. Dori, and J. Sussman. *COIM: An Object-Process Based Method for Analyzing Architectures of Complex, Interconnected, Large-Scale Socio-Technical Systems*. Systems Engineering 14(3), 2011.

Paredis, C.J.J., Y. Bernard, R.M. Burkhart, H-P. de Koning, S. Friedenthal, P. Fritzson, N.F. Rouquette, W. Schamai. 2010. "An overview of the SysML-modelica transformation specification". Proceedings of the 20th Annual International Council on Systems Engineering (INCOSE) International Symposium, 12-15 July 2010, Chicago, IL.

Reinhartz-Berger, I. and D. Dori. "OPM vs. UML—Experimenting with Comprehension and Construction of Web Application Models." *Empirical Software Engineering*, 10: 57–79, 2005.

Weilkiens, T. 2008. *Systems Engineering with SysML/UML*. Needham, MA, USA: OMG Press.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.amazon.com/gp/product/3540654712/sr=8-1/qid=1146053424/ref=sr_1_1/104-2484506-3323967?_encoding=UTF8
[2] http://esml.iem.technion.ac.il/?page_id=874

Knowledge Area: Systems Approach Applied to Engineered Systems

Systems Approach Applied to Engineered Systems

This knowledge area (KA) provides a guide for applying the systems approach as a means of identifying and understanding complex problems and opportunities, synthesizing possible alternatives, analyzing and selecting the best alternative, implementing and approving a solution, as well as deploying, using and sustaining engineered system solutions. The active participation of stakeholders during all the activities of the systems approach is the key to the success of the systems approach.

In an engineered system context, a systems approach is a holistic approach that spans the entire life of the system; however, it is usually applied in the development and operational/support life cycle stages. This knowledge area defines a systems approach using a common language and intellectual foundation to ensure that practical systems concepts, principles, patterns and tools are accessible to perform systems engineering (SE), as is discussed in the introduction to Part 2: Foundations of Systems Engineering.

Topics

Each part of the Guide to the SE Body of Knowledge (SEBoK) is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Overview of the Systems Approach
- Engineered System Context
- Identifying and Understanding Problems and Opportunities
- Synthesizing Possible Solutions
- Analysis and Selection between Alternative Solutions
- Implementing and Proving a Solution
- Deploying, Using, and Sustaining Systems to Solve Problems
- Stakeholder Responsibility
- Applying the Systems Approach

Systems Approach

This KA describes a high-level framework of activities and principles synthesized from the elements of the systems approach, as described earlier in Part 2 of the SEBoK, and is mapped to the articles Concepts of Systems Thinking, Principles of Systems Thinking, and Patterns of Systems Thinking. The concept map in Figure 1 describes how the knowledge is arranged in this KA and the linkage to the KA in Part 3.

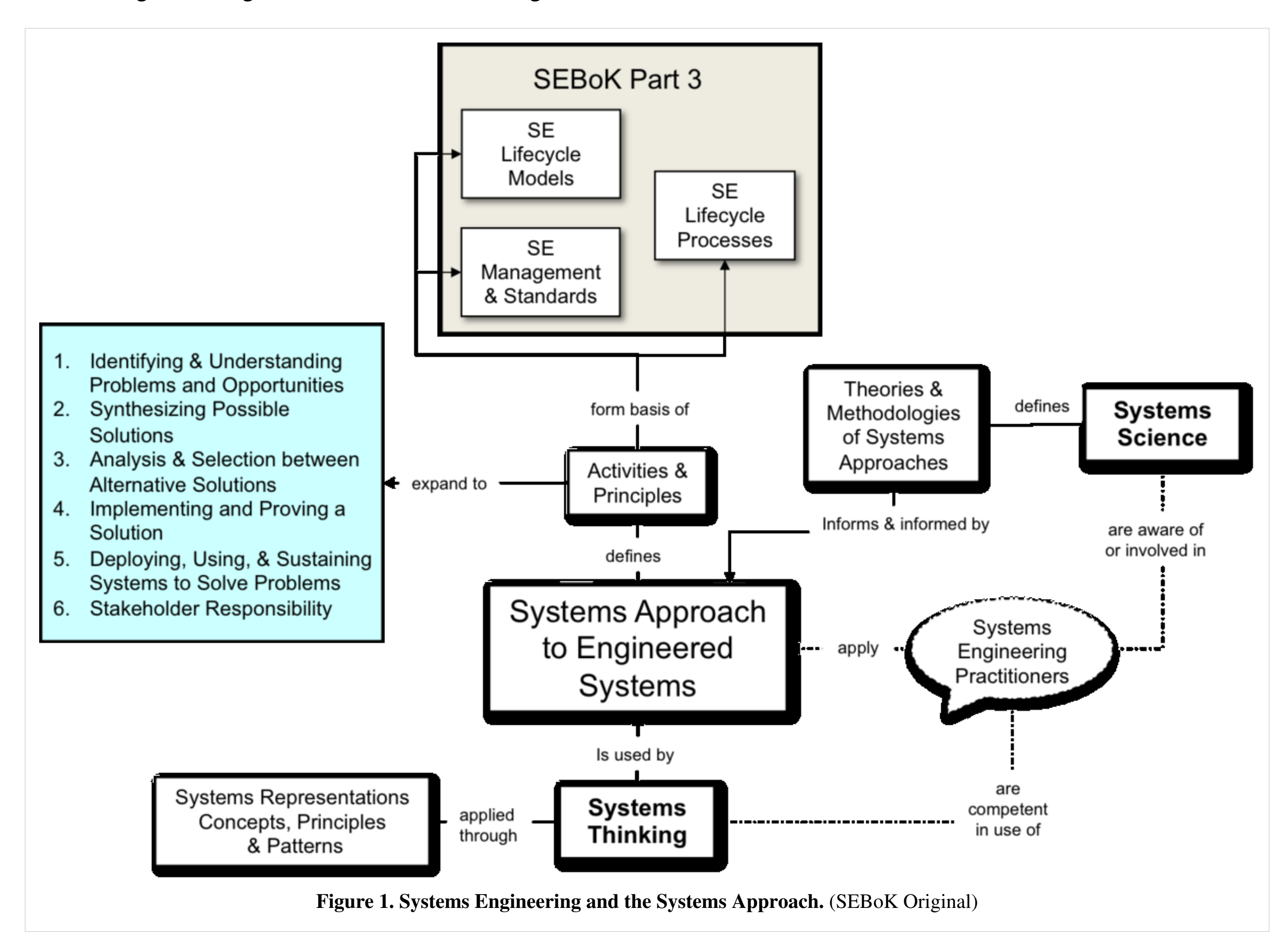


Figure 1. Systems Engineering and the Systems Approach. (SEBoK Original)

According to Jackson et al. (Jackson et al. 2010, 41-43), the systems approach to engineered systems is a problem-solving paradigm. It is a comprehensive problem identification and resolution approach based upon the principles, concepts, and tools of systems thinking and systems science, along with the concepts inherent in engineering problem-solving. It incorporates a holistic systems view that covers the larger context of the system, including engineering and operational environments, stakeholders, and the entire life cycle.

Successful systems practice should not only apply systems thinking to the system being created, but should also utilize systems thinking in consideration of the way in which work is planned and conducted. See Part 5: Enabling Systems Engineering for further discussions on how individuals, teams, businesses and enterprises may be enabled to perform systems engineering.

References

Works Cited

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?". INCOSE *Insight.* 13(1): 41-43.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1): 41-43.

Additional References

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology*. Hoboken, NJ, USA: John Wiley & Sons.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Senge, P. M. 1990. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York, Doubleday/Currency.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Overview of the Systems Approach

This knowledge area (KA) considers how a systems approach relates to engineered systems and to systems engineering (SE). The article Applying the Systems Approach considers the dynamic aspects of how the approach is used and how this relates to elements of SE.

Systems Approach and Systems Engineering

The term systems approach is used by systems science authors to describe a systems *"thinking"* approach, as it pertains to issues outside of the boundary of the immediate system-of-interest (Churchman 1979). This systems approach is essential when reductionist assumptions (the notion that the whole system has properties derived directly from the properties of their components) no longer apply to the system-of-interest (SoI) and when emergence and complexity at multiple levels of a system context necessitate a holistic approach.

The systems approach for engineered systems is designed to examine the *"whole system, whole lifecycle, and whole stakeholder community"* as well as to ensure that the purpose of the system (or systemic intervention) is achieved sustainably without causing any negative unintended consequences. This prevents the engineer from *"transferring the burden"* (in systems thinking terms) to some other part of the environment that unable to sustain that burden (Senge 2006). This also deters issues involving *sub-optimization* that could occur when whole systems are not kept in mind in achieving the purpose of the system (Sillitto 2012).

The systems approach (derived from systems thinking) and systems engineering (SE) have developed and matured, for the most part, independently; therefore, the systems science and the systems engineering communities differ in their views as to what extent SE is based on a systems approach and how well SE uses the concepts, principles, patterns and representations of systems thinking. These two views are discussed in the following sections.

Systems Science View

As discussed in the Systems Science article, some parts of the systems movement have been developed as a reaction to the perceived limitations of systems engineering (Checkland 1999).

According to Ryan (2008):

> ***Systems engineering has a history quite separate to the systems movement.*** *Its closest historical link comes from the application of systems analysis techniques to parts of the systems engineering process . . . The recent popularity of the SoS buzzword in the systems engineering literature has prompted* ***the expansion of systems engineering techniques to include methods that can cope with evolving networks of semi-autonomous systems. This has led many systems engineers to read more widely across the systems literature, and is providing a re-conceptualization of systems engineering as part of the systems movement, despite its historical independence. This is reflected in the latest INCOSE hand-book [INCOSE 2011, page 52], which states "the systems engineering perspective is based on systems thinking", which "recognizes circular causation, where a variable is both the cause and the effect of another and recognizes the primacy of interrelationships and non-linear and organic thinking—a way of thinking where the primacy of the whole is acknowledged.*** **(emphases added)**

Thus, for many in the systems science community, systems thinking is *not* naturally embedded in either SE definitions or practice.

Systems Engineering View

Many SE authors see a clear link between SE and systems thinking. For example, Hitchins (Hitchins 2007) describes generic models for the application of systems thinking to engineered system contexts. He suggests that these could form the foundation for descriptions and standards for the practices of SE. Hitchins also proposes a set of *guiding principles which have been the foundations of SE, apparently since its inception* (Hitchins 2009):

- **SE Principle A: The Systems Approach** - *"SE is applied to a system-of-interest (SoI) in a wider systems context"*
- **SE Principle B: Synthesis** - *"SE must bring together a collection of parts to create whole system solutions"*
- **SE Principle C: Holism** - *"Always consider the consequences on the wider system when making decisions about the system elements"*
- **SE Principle D: Organismic Analogy** - *"Always consider systems as having dynamic "living" behavior in their environment"*
- **SE Principle E: Adaptive Optimizing** - *"Solve problems progressively over time"*
- **SE Principle F: Progressive Entropy Reduction** - *"Continue to make systems work over time, through maintenance, sustainment, upgrade activities."*
- **SE Principle G: Adaptive Satisfying** - *"A system will succeed only if it makes winners of its success-critical stakeholders, so the lifecycle of a system must be driven by how well its outputs contribute to stakeholder purpose"*

Hitchins considers principles A-D as pillars of SE that identify key aspects of systems thinking which should underpin the practice of SE. Principles E-G consider the dynamics of SE life cycle thinking, the *why*, *when* and *how often* of SE.

The following sections consider the systems approach to engineered systems against four themes.

1. Whole System

The system coupling diagram (Figure 1), describes the scope of a systems approach to engineered systems (Lawson 2010).

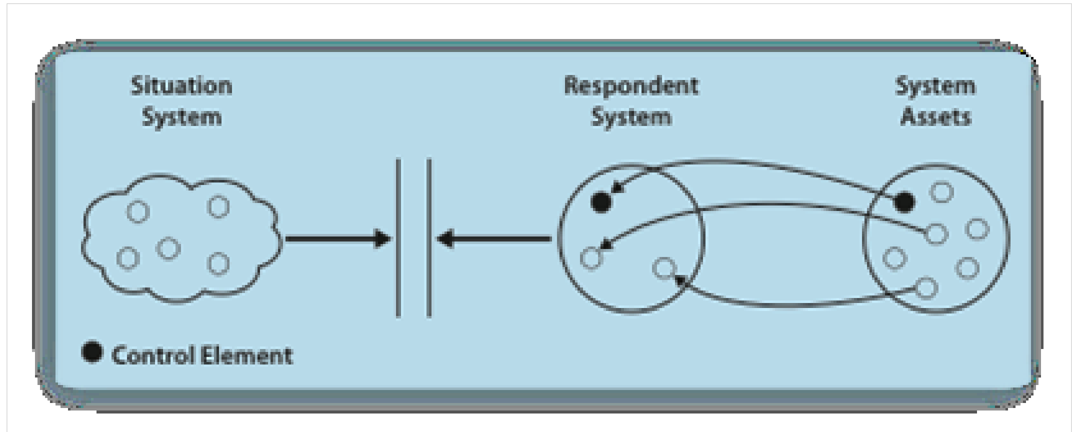


Figure 1. System Coupling Diagram (Lawson 2010). Reprinted with permission of Harold "Bud" Lawson.

- The **situation system** is the problem or opportunity either unplanned or planned. The situation may be natural, man-made, a combination of both, or a postulated situation used as a basis for deeper understanding and training (e.g. business games or military exercises).
- The **respondent system** is the system created to respond to the situation. The parallel bars indicate that this system interacts with the situation and transforms it to a new situation. Respondent systems have several names, project, program, mission, task force, or in a scientific context, experiment.
- **System assets** are the sustained assets of one or more enterprises to be used in response to situations. System assets must be adequately managed throughout the life of a system to ensure that they perform their function when instantiated in a respondent system. Examples include: value-added products or services, facilities, instruments and tools, and abstract systems, such as theories, knowledge, processes and methods.

Martin (Martin 2004) describes seven types of system, or *"the seven samurai of systems engineering"*, all of which, system developers need to understand to develop successful systems:

- the context system
- the intervention system
- the realization system
- the deployed system
- collaborating systems
- the sustainment system
- competing systems

Martin contends that all seven systems must be explicitly acknowledged and understood when engineering a solution for a complex adaptive situation.

These views, and others, describe one aspect of the systems approach when applied to engineered systems; in addition, it is applicable to understanding a problem, it organizes the resolution of that problem, and creates and integrates any relevant assets and capabilities to enable that solution.

2. Whole Lifecycle

Ring (Ring 1998) provides a powerful framework for the continuing management and periodic upgrade of long-life and *"immortal"* systems. It also accurately represents the *"continuous"* or very rapid product launch and refreshment cycle driven by market feedback and constant innovation that is seen in most product and service system consumer markets.

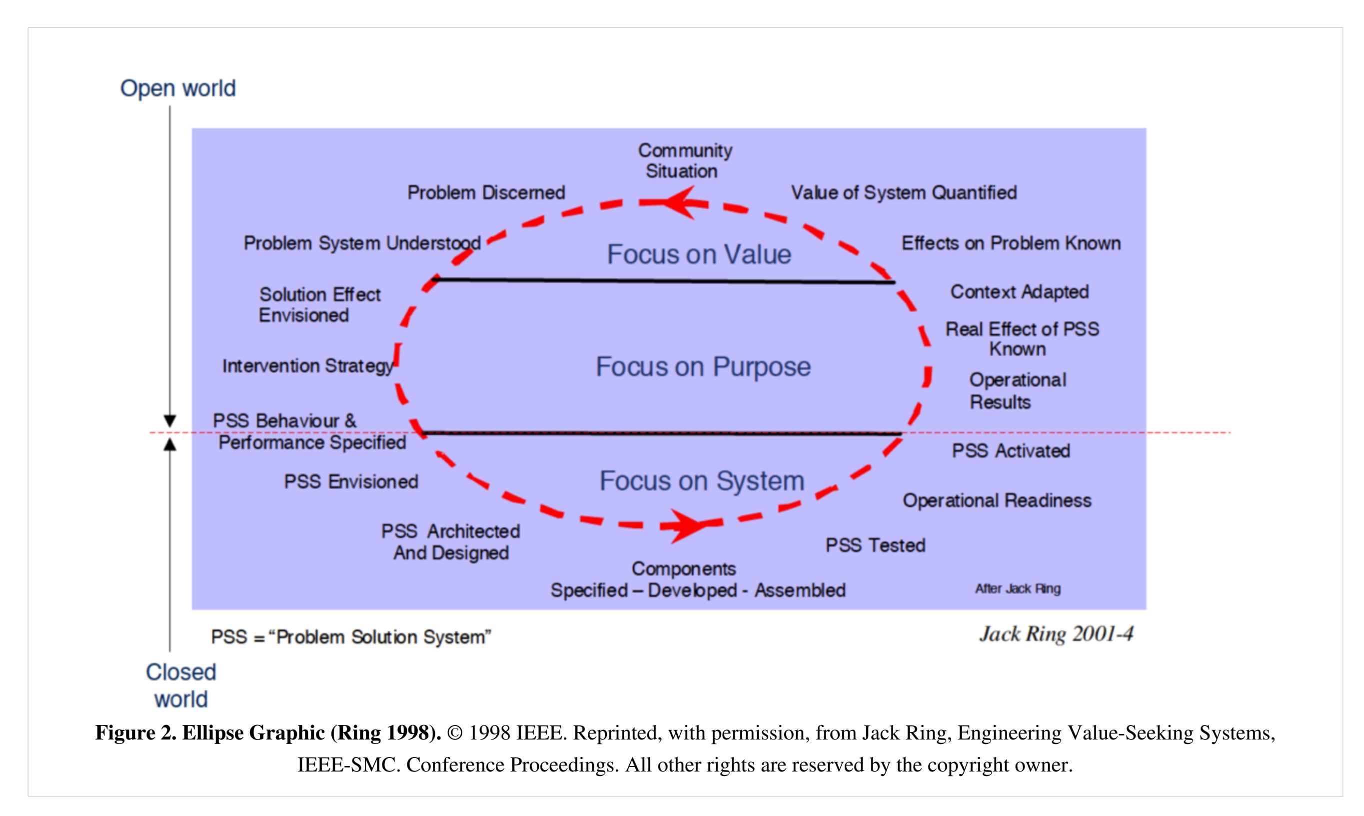


Figure 2. Ellipse Graphic (Ring 1998). © 1998 IEEE. Reprinted, with permission, from Jack Ring, Engineering Value-Seeking Systems, IEEE-SMC. Conference Proceedings. All other rights are reserved by the copyright owner.

Enterprise systems engineering may be considered in multiple concurrent instances of this model for different sub-sets of enterprise assets and services, in order to maintain a capability to pursue enterprise goals in a complex and dynamic external environment.

The dynamic nature of this cycle and its relationship to Life Cycle thinking is discussed in the article Applying the Systems Approach.

3. Whole Problem

The article Identifying and Understanding Problems and Opportunities considers the nature of problem situations. It discusses the relationship between hard system and soft system views of problems and how they relate to engineered systems. Engineered systems are designed to operate with and add value to a containing social and/or ecological system. The scope of problems is captured by frameworks, such as Political, Economic, Social, Technological, Legal and Environmental (PESTLE) (Gillespie 2007) or Social, Technical, Economic, Environmental, Political, Legal, Ethical and Demographic (STEEPLED).

The idea of a wicked problem (Rittel and Webber 1973) is also discussed. These problems cannot be quantified and solved in a traditional engineering sense.

Sillitto (Sillitto 2010) describes a lifecycle model in which the decision as to what parts of problems can be *"solved"* and what parts must be *"managed"* is the first key decision and emphasizes the need for a solution approach that provides flexibility in the solution to match the level of uncertainty and change in the problem and stakeholder expectations. It is now normal to view a problem as one that *"changes over time"* and to promote the belief that value is determined by the perceptions of key stakeholders.

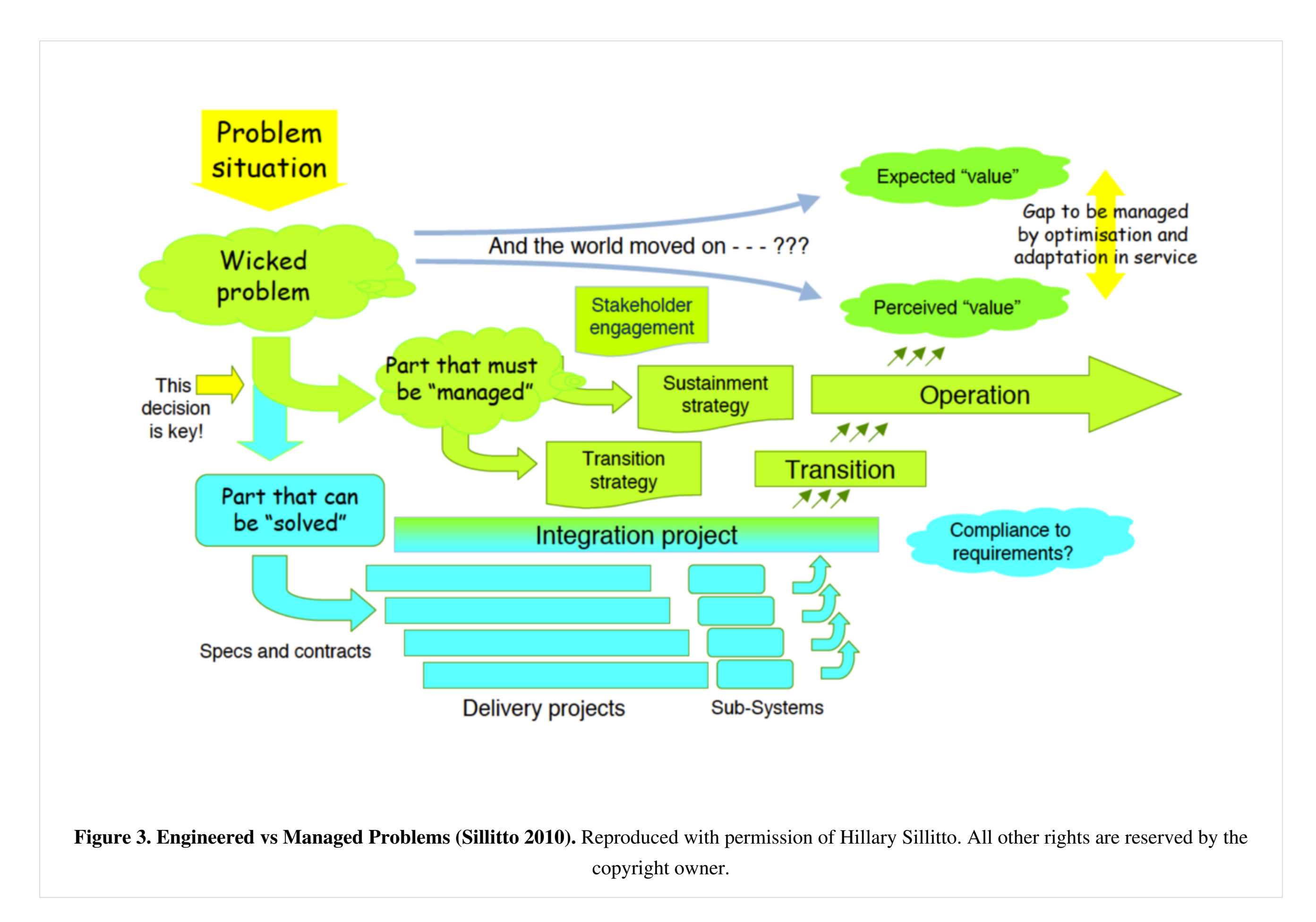


Figure 3. Engineered vs Managed Problems (Sillitto 2010). Reproduced with permission of Hillary Sillitto. All other rights are reserved by the copyright owner.

Thus, a systems approach can be useful when addressing all levels of a problematic situation, from individual technologies to the complex socio-technical issues that come about in the area of engineered systems development.

4. Multi-Disciplinary

As discussed by Sillitto (Sillitto 2012), the methods and thinking applied by many practicing systems engineers have become optimized to the domains of practice. While systems thinking concepts, patterns and methods are used widely, they are not endemic in SE practice. As a result, SE practitioners find it difficult to share systems ideas with others involved in a systems approach. Part 4: Applications of Systems Engineering describes traditional (product based) SE (Lawson 2010) and examines this against the SE approaches that are applicable to service, enterprise, and system of systems capability. These approaches require more use of problem exploration, a broader solution context, and a purpose driven life cycle thinking.

SE and Systems Approach

From the above discussions, there are three ways in which SE could make use of a systems approach:

- in its overall problem solving approach
- in the scope of problem and solution system contexts considered
- in the embedding of systems thinking and systems thinking tools and in all aspects of the conduct of that approach

The current SE standards and guides, as described in Part 3: Systems Engineering and Management, encapsulate many of the elements of a systems approach. However, they tend to focus primarily on the development of system solutions while the wider purpose-driven thinking of a full systems approach (Ring 1998) and the wider consideration of all relevant systems (Martin 2004) are embedded in the acquisition and operational practices of their application domains.

The inclusion of systems thinking in SE competency frameworks (INCOSE 2010) represents a general move toward a desire for more use of systems thinking in SE practice. There is a wide stakeholder desire to acquire the benefits of a systems approach through the application of SE, particularly in areas where current SE approaches are inadequate or irrelevant. Hence, there is a need for a better articulation of the *systems approach* and how to apply it to non-traditional problems.

Synthesis for SEBOK

The systems approach presented in the SEBoK uses the following activities:

- identify and understand the relationships between the potential problems and opportunities in a real world situation
- gain a thorough understanding of the problem and describe a selected problem or opportunity in the context of its wider system and its environment
- synthesize viable system solutions to a selected problem or opportunity situation
- analyze and choose between alternative solutions for a given time/cost/quality version of the problem.
- provide evidence that a solution has been correctly implemented and integrated
- deploy, sustain, and apply a solution to help solve the problem (or exploit the opportunity)

All of the above are considered within a life cycle (glossary) framework which may need concurrent, recursive (glossary) and iterative applications of some or all of the systems approach.

When the systems approach is executed in the real world of an engineered system (glossary), a number of engineering and management disciplines emerge, including SE. Part 3: Systems Engineering and Management and Part 4: Applications of Systems Engineering contain a detailed guide to SE with references to the principles of the systems approach, where they are relevant. Part 5: Enabling Systems Engineering provides a guide to the relationships between SE and the organizations and Part 6: Related Disciplines also offers a guide to the relationship between SE and other disciplines.

More detailed discussion of how the systems approach relates to these engineering and management disciplines is included in the article Applying the Systems Approach within this KA.

References

Works Cited

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Churchman, C.W. 1979. *The Systems Approach and Its Enemies.* New York, NY: Basic Books.

Gillespie. 2007. *Foundations of Economics - Additional chapter on Business Strategy.* Oxford University Press.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?". INCOSE *Insight*. 12(4).

Hitchins, D. 2007. *Systems Engineering, a 21st Century Systems Methodology.* Hoboken, NJ: Wiley.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205.* San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Martin, J, 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems." Proceedings of the 14th Annual INCOSE International Symposium, 20-24 June 2004, Toulouse, France.

Ring, J., 1998. "A Value Seeking Approach to the Engineering of Systems." Proceedings of the IEEE Conference on Systems, Man, and Cybernetics. p. 2704-2708.

Rittel, H. and M. Webber. 1973. "Dilemmas in a General Theory of Planning." *Policy Sciences*. 4:155–169.

Ryan, A. 2008. “What is a Systems Approach?” *Journal of Non-linear Science*.

Senge, P.M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*, 2nd ed. New York, NY, USA: Doubleday Currency.

Sillitto, H. 2010. “Design principles for ultra-large scale systems.” Proceedings of the INCOSE International Symposium, Chicago, July 2010, re-printed in "The Singapore Engineer" July 2011.

Sillitto, H.G. 2012: "Integrating Systems Science, Systems Thinking, and Systems Engineering: understanding the differences and exploiting the synergies", Proceedings of the INCOSE International Symposium, Rome July 2012.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight*. 12(4).

Senge, P.M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*, 2nd ed. New York, NY, USA: Doubleday Currency.

Additional References

Biggs, J.B. 1993. "From Theory to Practice: A Cognitive Systems Approach". *Journal of Higher Education & Development.* Accessed December 4 2014 Taylor and Francis from http:/ / www. informaworld. com/ smpp/ content~db=all~content=a758503083.

Boardman, J. and B. Sauser 2008. *Systems Thinking: Coping with 21st Century Problems*. Boca Raton, FL, USA: CRC Press.

Edson, R. 2008. *Systems Thinking. Applied. A Primer.* Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

Ring J. 2004. "Seeing an Enterprise as a System." INCOSE *Insight.* 6(2): 7-8.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Engineered System Context

This article is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the further expansion of the ideas of an engineered system and engineered system context that were introduced in the Systems Fundamentals KA.

The single most important principle of the systems approach is that it is applied to an engineered system context and not just to a single system (INCOSE 2012). The systems approach includes models and activities useful for the understanding, creation, use, and sustainment of engineered systems to enable the realization of stakeholder needs. Disciplines that use a systems approach (like systems engineering (SE)) consider an engineered system context that defines stakeholder needs, and look for the best ways to provide value by applying managed technical activities to one or more selected engineered systems of interest (SoI).

Generally, four specific types of engineered system contexts are recognized in SE:

- product system (glossary)
- service system (glossary)
- enterprise system (glossary)
- system of systems (SoS) capability

One of the key distinctions between these system contexts pertains to the establishment of how and when the SoI boundary is drawn.

Engineered System-of-Interest

We use the idea of an engineered system context to define an engineered SoI and to capture and agree on the important relationships between it, the systems it works directly with, and any other systems with which it work. All applications of a systems approach (and hence of SE) are applied in a system context rather than only to an individual system.

A system context can be constructed around the following set of open system relationships (Flood and Carson 1993):

- The **Narrower System-of-Interest** (NSoI) is the system of direct concern to the observer. The focus of this system is driven by the scope of authority or control with implicit recognition that this scope may not capture all related elements.
- The **Wider System-of-Interest** (WSoI) describes a logical system boundary containing all of the elements needed to fully understand system behavior. The observer may not have authority over all of the elements in the WSoI, but will be able to establish the relationships between WSoI elements and NSoI elements.
- The WSoI exists in an **environment**. The immediate environment contains engineered, natural, and/or social systems, with which the WSoI (and thus some elements of the NSoI) directly interact with for the purpose of exchanging material, information, and/or energy to achieve its goals or objective.
- A **Wider Environment** completes the context and contains systems that have no direct interaction with the SoI, but which might influence decisions related to it during its life cycle.
- "Some Theoretical Considerations of Mathematical Modeling" (Flood 1987) extends this context to include a **meta-system** (MS) that exists outside of the WSoI and exercises direct control over it.

The choice of the SoI boundary for particular activities depends upon what can be changed and what must remain fixed. The SoI will always include one or more NSoI, but may also include WSoI and a MS if appropriate, such as when considering a service or an enterprise system.

Applying the System Context

For lower-level and less-complex systems, the WSoI can represent levels of a product system hierarchy. An example of this would be an engine management unit as part of an engine, or an engine as part of a car. The WSoI in a system context may encapsulate some aspects of SoS ideas for sufficiently complex systems. In these cases, the WSoI represents a collection of systems with their own objectives and ownership with which the NSoI must cooperate with in working towards a shared goal. An example of this would be a car and a driver contributing to a transportation service.

This view of a SoS context being used as a means to support the engineering of a NSoI product system is one way in which a systems approach can be applied. It can also be applied directly to the SoS. Examples of this include a flexible multi-vehicle transportation service or transportation as part of a commercial enterprise. In this case, the NSoI aspect of the context no longer applies. The WSoI will consist of a set of cooperating systems, each of which might be changed or replaced to aid in the synthesis of a solution. The context may also need to represent **loose coupling**, with some systems moving in or out of the context depending on the need, or **late binding** with systems joining the context only at, or close to the delivery of the service.

Thus, a context allows a reductionist view of the SoI that is of direct concern to an observer, as it provides for the system relationships and influences that are needed to maintain a holistic (glossary) view of the consequence of any actions taken.

Product System Context

The distinction between a product (glossary) and a product system (glossary) is discussed in the article Types of Systems.

A product system context would be one in which the SoI is the product itself. The wider system context for a product system can be a higher level of product hierarchy, a service, or an enterprise system that uses the product directly to help provide value to the user. A significant aspect of a product systems context is the clear statement of how the product is intended to be used and ensures that this information is given to the acquirer upon delivery. The customer will be required to accept the system, typically through a formal process, agreeing not to go against the terms of use.

If a systems approach is applied to a product context, it is done with the purpose of engineering a narrow system product to be integrated and used in a wider system product hierarchy or to enable the delivery of a wider system service directly to a user by an enterprise.

This view of the relationship between product and service is specific to product systems engineering. While some engineering of the acquirer's static service system may occur, it is done with a product focus. The definition of service system in a service systems engineering context describes a more dynamic view of service systems.

Service System Context

Services are activities that cause a transformation of the state of an entity (people, product, business, and region or nation) by mutually agreed terms between the service provider and the customer (Spohrer 2008). The distinction between service and a service system is discussed in the article Types of Systems.

A service system context is one in which the SoI is the service system. This SoI contains all of the technology, infrastructure, people, resources, etc. that are needed to enable the service. The WSoI describes the enterprise providing the service as well as its relationship with other services that are impact the success of the enterprise.

If a systems approach is applied to a service system, it is done with the purpose of engineering a service system to enable the outcomes required by an enterprise to satisfy its clients. When operating in the service system context, all options to provide the service must be considered, providing that they fit within the constraints of the enterprise. This will include interfaces to other services, people, and resources in the enterprise. If an option for providing the service

makes use of existing products or resources within or outside of the enterprise, it must be ensured that they are available for this use and that this does not adversely affect other services. Part of getting the right service may require the negotiation of changes to the wider enterprise context, but this must be by agreement with the relevant authority.

For a service system, and also when considering the service system context, the value is realized only through service transactions. The end-user co-creates value at the time of the request to use the service. For example, to make a flight reservation using a smart phone, the service system is composed of many service system entities (the caller, the person called, the smart phone, the access network, the core Internet Protocol (IP) network, the Internet Service provider (ISP), the World Wide Web (WWW), data centers, etc. All these are necessary to enable the service. When a caller makes a reservation and then books the flight, the value has been created.

This definition of a service system, as associated with dynamic Information Technology (IT) services, is discussed further in the article Service Systems Engineering.

Enterprise System Context

The distinction between an enterprise and an enterprise system is discussed in the article Types of Systems.

An enterprise system context is one in which the SoI is the enterprise system. This system contains all of the technology, infrastructure, people, resources, etc. needed to enable the service. The WSoI describes the business environment within which the enterprise sits.

It is to be noted that an enterprise context is not equivalent to an **organization** according to this definition. An enterprise includes not only the organizations that participate in it, but also the people, knowledge, and other assets, such as processes, principles, policies, practices, doctrines, theories, beliefs, facilities, land, and intellectual property that compose the enterprise.

An enterprise may contain or employ service systems along with product systems. An enterprise might even contain sub-enterprises. Enterprise systems are unique when compared to product and service systems in that:

- they are constantly evolving
- they rarely have detailed configuration controlled requirements
- they typically have (constantly changing) goals of providing shareholder value and customer satisfaction
- they exist in a context (or environment) that is ill-defined and constantly changing

The enterprise systems engineer must consider and account for these factors in their processes and methods.

Both product and service systems require an enterprise system context to create them and an enterprise to use the product system and deliver services, either internally to the enterprise or externally to a broader community. Thus, the three types of engineered system contexts are linked in all instances, regardless of which type of system the developers consider as the object of the development effort that is delivered to the customer.

References

Works Cited

Flood, R.L. 1987. "Some Theoretical Considerations of Mathematical Modeling." In Problems of Constancy and Change (proceedings of 31st Conference of the International Society for General Systems Research, Budapest), Volume 1, p. 354 - 360.

Flood, R.L. and E.R. Carson. 1993. Dealing with Complexity: An Introduction to the Theory and Application of Systems Science, 2nd ed. New York, NY, USA: Plenum Press.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE),

INCOSE-TP-2003-002-03.2.2.

Spohrer, J. 2008. "Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline-Outline & References." *International Journal of Information Systems in the Service Sector.* 1(3) (May).

Primary References

Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice.* Boca Raton, FL, USA: CRC Press.

Rouse, W.B. 2005. "Enterprise as Systems: Essential Challenges and Enterprise Transformation". *Systems Engineering*. 8(2): 138-50.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

Additional References

ANSI/EIA. 2003. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA). ANSI/EIA 632-1998.

Bernus, P., L. Nemes, and G. Schmidt (eds.). 2003. *Handbook on Enterprise Architecture.* Heidelberg, Germany: Springer.

Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and Sons.

DeRosa, J. K. 2006. “An Enterprise Systems Engineering Model.” Proceedings of the 16th Annual International Council on Systems Engineering, 9-13 July 2006, Orlando, FL, USA.

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods.* Boca Raton, FL, USA: CRC Press.

Joannou, P. 2007. "Enterprise, Systems, and Software—The Need for Integration." IEEE *Computer*. 40(5) (May): 103-105.

Katzan, H. 2008. *Service Science.* Bloomington, IN, USA: iUniverse Books.

Maglio P., S. Srinivasan, J.T. Kreulen, and J. Spohrer. 2006. “Service Systems, Service Scientists, SSME, and Innovation." *Communications of the ACM*. 49(7) (July).

Martin J.N. 1997. *Systems Engineering Guidebook.* Boca Raton, FL, USA: CRC Press.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice.* Boca Raton, FL, USA: CRC Press.

Rouse, W.B. 2009. "Engineering the Enterprise as a System". In *Handbook of Systems Engineering and Management*, 2nd ed. A.P. Sage and W.B. Rouse (eds.). New York, NY, USA: Wiley and Sons.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

Valerdi, R. and D.J. Nightingale. 2011. "An Introduction to the Journal of Enterprise Transformation." *Journal of Enterprise Transformation* 1(1):1-6.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Identifying and Understanding Problems and Opportunities

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the identification and exploration of problems or opportunities in detail. The problem situations described by the activities in this topic may form a starting point for Synthesizing Possible Solutions. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this knowledge area, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

The phrase "problem or opportunity" used herein recognizes that the "problem" is not always a negative situation and can also be a positive opportunity to improve a situation.

Introduction

According to Jenkins (1969), the first step in the systems approach is “the recognition and formulation of the problem.” The systems approach described in the Guide to the SE Body of Knowledge (SEBoK) is predominantly a hard system approach. The analysis, synthesis, and proving parts of the approach assume a problem or opportunity has been identified and agreed upon and that a "new" engineered system (glossary) solution is needed.

However, the systems approach does not have to apply to the development and use of a newly designed and built technical solution. Abstract or experimental solutions to potential problems might be explored to help achieve agreement on a problem context. Solutions may involve reorganizing existing system of systems (SoS) contexts or the modification or re-use of existing products and services. The problem and opportunity parts of the approach overlap with soft system approaches. This is discussed in more detail below.

One thing that must be considered in relation to system complexity is that the opportunity situation may be difficult to fully understand; therefore, system solutions may not solve the problem the first time, but is still useful in increasing the understanding of both problem issues and what to try next to work toward a solution.

Hence, problem exploration and identification is often not a one-time process that specifies the problem, but is used in combination with solution synthesis and analysis to progress toward a more complete understanding of problems and solutions over time (see Applying the Systems Approach for a more complete discussion of the dynamics of this aspect of the approach).

Problem Exploration

Soft system thinking does not look for "the problem", but considers a problematic situation. Forming systems views of this situation can help stakeholders better understand each other's viewpoints and provide a starting point for directed intervention in the current system context. If a full soft systems intervention is undertaken, such as a soft systems methodology (SSM) (Checkland 1999), it will not include formal analysis, synthesis, and proving. However, the SSM method was originally based on hard methodologies, in particular one presented by Jenkins (1969). It follows the basic principles of a systems approach: "analyzing" conceptual models of shared understanding,

"synthesizing" intervention strategies, and "proving" improvements in the problematic situation.

Often, the distinction between hard and soft methods is not as clear cut as the theory might suggest. Checkland himself has been involved in applications of SSM as part of the development of information system design (Checkland and Holwell 1998). It is now agreed upon by many that while there is a role for a "pure soft system" approach, the service (glossary) and enterprise (glossary) problems now being tackled can only be dealt with successfully by a combination of soft problematic models and hard system solutions. Mingers and White (Mingers and White 2009) give a number of relevant examples of this. In particular they reference "Process and Content: Two Ways of Using SSM" (Checkland and Winters 2006). It is likely in the future that engineered system problems will be stated, solved, and used as part of a predominately soft intervention, which will place pressure on the speed of development needed in the solution space. This is discussed more fully in the topic Life Cycle Models.

The critical systems thinking and multi-methodology approaches (Jackson 1985) take this further by advocating a "pick and mix" approach, in which the most appropriate models and techniques are chosen to fit the problem rather than following a single methodology (Mingers and Gill 1997). Thus, even if the hard problem identification approach described below is used, some use of the soft system techniques (such as rich pictures, root definitions, or conceptual models) should be considered within it.

Problem Identification

Hard system thinking is based on the premise that a problem exists and can be stated by one or more stakeholders in an objective way. This does not mean that hard systems approaches start with a defined problem. Exploring the potential problem with key stakeholders is still an important part of the approach.

According to Blanchard and Fabrycky (Blanchard and Fabrycky 2006, 55-56), defining a problem is sometimes the most important and difficult step. In short, a system cannot be defined unless it is possible to clearly describe what it is supposed to accomplish.

According to Edson (Edson 2008, 26-29), there are three kinds of questions that need to be asked to ensure we fully understand a problem situation. First, how difficult or well understood is the problem? The answer to this question will help define the tractability of the problem. Problems can be “tame,” “regular,” or “wicked”:

- For tame problems, the solution may be well-defined and obvious.
- Regular problems are those that are encountered on a regular basis. Their solutions may not be obvious, thus serious attention should be given to every aspect of them.
- Wicked problems (Rittel and Webber 1973) cannot be fully solved, or perhaps even fully defined. Additionally, with wicked problems, it is not possible to understand the full effect of applying systems to the problem.

Next, who or what is impacted? There may be elements of the situation that are causing the problem, elements that are impacted by the problem, and elements that are just in the loop. Beyond these factors, what is the environment and what are the external factors that affect the problem? In examining these aspects, the tools and methods of systems thinking can be productively applied.

Finally, what are the various viewpoints of the problem? Does everyone think it is a problem? Perhaps there are conflicting viewpoints. All these viewpoints need to be defined. Persons affected by the system, who stand to benefit from the system, or can be harmed by the system, are called stakeholders. Wasson (Wasson 2006, 42-45) provides a comprehensive list of stakeholder types. The use of soft systems models, as discussed above, can play an important part in this. Describing a problem using situation views can be useful when considering these issues, even if a single problem perspective is selected for further consideration.

Operations research is a hard systems method which concentrates on solving problem situations by deploying known solutions. The problem analysis step of a typical approach asks questions about the limitation and cost of the current system to identify efficiency improvements that need to be made (Flood and Carson 1993).

Traditional SE methods tend to focus more on describing an abstract model of the problem, which is then used to develop a solution that will produce the benefits stakeholders expect to see (Jenkins 1969). The expectation is often that a new solution must be created, although this need not be the case. Jenkins suggests that SE is just as applicable to a redesign of existing systems. A clear understanding of stakeholder expectations in this regard should produce a better understanding of part of the problem. Do stakeholders expect a new solution or modifications to their existing solutions, or are they genuinely open to solution alternatives which consider the pros and cons of either. Such expectations will influence suggestions of solution alternatives, as discussed in the Synthesizing Possible Solutions article.

An important factor in defining the desired stakeholder outcomes, benefits, and constraints is the operational environment, or scenario, in which the problem or opportunity exists. Armstrong (Armstrong 2009, 1030) suggests two scenarios: the first is the descriptive scenario, or the situation as it exists now, and the second is the normative scenario, or the situation as it may exist sometime in the future.

All of these aspects of problem understanding can be related to the concept of a system context.

Problem Context

The Engineered System Context topic identifies a way by which a complex system situation can be resolved around a system-of-interest (glossary) (SoI). The initial identification of a "problem context" can be considered as the outcome of this part of the systems approach.

The systems approach should not consider only soft or hard situations. More appropriately, a problem or opportunity should be explored using aspects of both. In general, the application of the systems approach with a focus on engineered system contexts will lead to hard system contexts in which an identified SoI and required outcome can be defined.

An initial description of the wider SoI and environment serves as the problem or opportunity problem scope. Desired stakeholder benefits are expressed as outcomes in the wider system and some initial expression of what the SoI is intended for may be identified. Jenkins (1969) defines a problem formulation approach where one:

- states the aim of the SoI
- defines the wider SoI
- defines the objectives of the wider SoI
- defines the objectives of the system
- defines economic, informational, and other conditions.

In a hard system problem context, a description of a logical or ideal system solution may be included. This ideal system cannot be implemented directly, but describes the properties required of any realizable system solution.

To support this problem or opportunity description, a soft context view of the SoI will help ensure wider stakeholder concerns are considered. If a soft system context has been defined, it may include a conceptual model (Checkland 1999) which describes the logical elements of a system that resolve the problem situation and how they are perceived by different stakeholders. Unlike the hard system view, this does not describe the ideal solution, but provides an alternative view on how aspects of any solution would be viewed by potential stakeholders.

In problem contexts with a strong coercive dimension, the problem context should include an identification of the relative power and the importance of stakeholders.

The problem context should include some boundaries on the cost, time to deployment, time in use, and operational effectiveness needed by stakeholders. In general, both the full problem context and an agreed version of the problem to be tackled next are described. (see Applying the Systems Approach).

References

Works Cited

Armstrong, Jr., J.E., 2009. "Issue Formulation." In A.P. Sage and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management,* 2nd edition. Hoboken, NJ, USA: Wiley.

Blanchard, B. and W.J. Fabrycky. 2006. *Systems Engineering and Analysis*. Upper Saddle River, NJ, USA: Prentice Hall.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: Wiley.

Checkland, P. and S. Holwell. 1998. *Information, Systems and Information Systems: Making Sense of the Field*. Hoboken, NJ, USA: Wiley.

Checkland, P. and M. Winter. 2006. "Process and Content: Two Ways of Using SSM". *Journal of Operational Research Society.* 57(12): 1435-1441.

Edson, R. 2008. *Systems Thinking. Applied. A Primer*. Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

Flood, R. L. and E.R. Carson 1993. *Dealing with Complexity: An Introduction to the Theory and Application of Systems Science,* 2nd ed. New York, NY, USA: Plenum Press.

Jackson, M. 1985. "Social Systems Theory and Practice: The Need for A Critical Approach". *International Journal of General Systems*. 10: 135-151.

Jenkins, G.M. 1969. "The Systems Approach." *The Journal of Systems Engineering.* 1(1).

Mingers, J. and A. Gill. 1997. *Multimethodology: Theory and Practice of Combining Management Science Methodologies*. Chichester, UK: Wiley.

Mingers, J. and L. White. 2009. *A Review of Recent Contributions of Systems Thinking to Operational Research and Management Science*, Working Paper 197. Canterbury, UK: Kent Business School.

Rittel, H. and M. Webber. 1973. "Dilemmas in a General Theory of Planning." In *Policy Sciences*. 4:155–169.

Wasson, C.S. 2006. *System Analysis, Design, and Development.* Hoboken, NJ, USA: Wiley.

Primary References

Blanchard, B. & W.J. Fabrycky. 2006. *Systems Engineering and Analysis*. Upper Saddle River, NJ, USA: Prentice Hall.

Edson, R. 2008. *Systems Thinking. Applied. A Primer*. Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Synthesizing Possible Solutions

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the synthesis of possible solution options in response to the problem situations described by activities from Identifying and Understanding Problems and Opportunities topic. The solution options proposed by the synthesis activities will form the starting point for the Analysis and Selection between Alternative Solutions. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Synthesis Overview

System synthesis is an activity within the systems approach that is used to describe one or more system solutions based upon a problem context for the life cycle of the system to:

- Define options for a SoI with the required properties and behavior for an identified problem or opportunity context.
- Provide solution options relevant to the SoI in its intended environment, such that the options can be assessed to be potentially realizable within a prescribed time limit, cost, and risk described in the problem context.
- Assess the properties and behavior of each candidate solution in its wider system context.

The iterative activity of system synthesis develops possible solutions and may make some overall judgment regarding the feasibility of said solutions. The detailed judgment on whether a solution is suitable for a given iteration of the systems approach is made using the Analysis and Selection between Alternative Solutions activities.

Essential to synthesis is the concept of holism, according to Hitchins (Hitchins 2009), and states that a system must be considered as a whole and not simply as a collection of its elements. The holism of any potential solution system requires that the behavior of the whole be determined by addressing a system within an intended environment and not simply the accumulation of the properties of the elements. The latter process is known as reductionism and is the opposite of holism, which Hitchins (Hitchins 2009, 60) describes as the notion that “the properties, capabilities, and behavior of a system derive from its parts, from interactions between those parts, and from interactions with other systems.”

When the system is considered as a whole, properties called emergent properties often appear (see Emergence). These properties are often difficult to predict from the properties of the elements alone. They must be evaluated within the systems approach to determine the complete set of performance levels of the system. According to Jackson (Jackson 2010), these properties can be considered in the design of a system, but to do so, an iterative approach is required.

In complex systems, individual elements will adapt to the behavior of the other elements and to the system as a whole. The entire collection of elements will behave as an organic whole. Therefore, the entire synthesis activity, particularly in complex systems, must itself be adaptive.

Hence, synthesis is often not a one-time process of solution design, but is used in combination with problem understanding and solution analysis to progress towards a more complete understanding of problems and solutions over time (see Applying the Systems Approach topic for a more complete discussion of the dynamics of this aspect of the approach).

Problem or Opportunity Context

System synthesis needs the problem or opportunity that the system is intended to address to have already been identified and described and for non-trivial systems, the problem or opportunity needs to be identified and understood concurrently with solution synthesis activities.

As discussed in Identifying and Understanding Problems and Opportunities, the systems approach should not consider strictly soft or hard situations. In general, the application of the systems approach, with a focus on engineered system contexts, will lead to hard system contexts in which an identified SoI and required outcome be defined. Even in these cases, a soft context view of the SoI context will help ensure wider stakeholder concerns are considered.

The problem context should include some boundaries on the cost, time to deployment, time in use, and operational effectiveness needed by stakeholders. In general, the goal is not to synthesize the perfect solution to a problem, but rather to find the best available solution for the agreed version of the problem.

Synthesis Activities

The following activities provide an outline for defining the SoI: grouping of elements, identification of the interactions among the elements, identification of interfaces between elements, identification of external interfaces to the SoI boundary and common sub-elements within the SoI boundary.

The activities of systems synthesis are built on the idea of a balanced reduction vs. holism approach as discussed in What is Systems Thinking? topic. It is necessary to divide system elements and functions to create a description of the SoI which is realizable, either through combinations of available elements or through the design and construction of new elements. However, if the system is simply decomposed into smaller and smaller elements, the holistic nature of systems will make it more and more difficult to predict the function and behavior of the whole. Thus, synthesis progresses through activities that divide, group, and allocate elements, and then assesses the complete system’s properties in context relevant to the user need the SoI will fulfill. Hence, synthesis occurs over the entire life cycle of the system as the system and its environment change.

Identification of the Boundary of a System

Establishing the boundary of a system is essential to synthesis, the determination of the system's interaction with its environment and with other systems, and the extent of the SoI. Buede (Buede 2009, 1102) provides a comprehensive discussion of the importance of, and methods of, defining the boundary of a system in a SE context.

Identification of the Functions of the System

The function of a system at a given level of abstraction is critical to synthesis since the primary goal of the synthesis activity is to propose realizable system descriptions which can provide a given function. The function of a system is distinct from its behavior as it describes what the system can be used for or is asked to do in a larger system context.

Buede (Buede 2009, 1091-1126) provides a comprehensive description of functional analysis in a SE context.

Identification of the Elements of a System

System synthesis calls for the identification of the elements of a system. Typical elements of an Engineered System Context may be physical, conceptual, or processes. Physical elements may be hardware, software, or humans. Conceptual elements may be ideas, plans, concepts, or hypotheses. Processes may be mental, mental-motor (writing, drawing, etc.), mechanical, or electronic (Blanchard and Fabrycky 2006, 7).

In addition to the elements of the system under consideration (i.e., a SoI), ISO 15288 (ISO/IEC/IEEE 15288 2015) also calls for the identification of the enabling systems. These are systems (or services) utilized at various stages in the life cycle, e.g., development, utilization or support stages, to facilitate the SoI in achieving its objectives.

Today's systems often include existing elements. It is rare to find a true "greenfield" system, in which the developers can specify and implement all new elements from scratch. "Brownfield" systems, wherein legacy elements constrain the system structure, capabilities, technology choices, and other aspects of implementation, are much more typical (Boehm 2009).

Division of System Elements

System synthesis may require elements to be divided into smaller elements. The division of elements into smaller elements allows the systems to be grouped and leads to the SE concept of physical architecture, as described by Levin (Levin 2009, 493-495). Each layer of division leads to another layer of the hierarchical view of a system. As Levin points out, there are many ways to depict the physical architecture, including the use of wiring diagrams, block diagrams, etc. All of these views depend on arranging the elements and dividing them into smaller elements. According to the principle of recursion, these decomposed elements are either terminal elements, or are able to be further decomposed. The hierarchical view does not imply a top-down analytical approach to defining a system. It is simply a view. In the systems approach, levels of the hierarchy are defined and considered recursively with one level forming the context for the next.

Grouping of System Elements

System synthesis may require that elements be grouped. This leads to the identification of the sub-systems that are essential to the definition of a system. Synthesis determines how a system may be partitioned and how each sub-system fits and functions within the whole system. The largest group is the SoI, also called the relevant system by Checkland (Checkland 1999, 166). According to Hitchins, some of the properties of a SoI are as follows: the SoI is open and dynamic, the SoI interacts with other systems, and the SoI contains sub-systems (Hitchins 2009, 61). The SoI is brought together through the concept of synthesis.

Identification of the Interactions among System Elements

System synthesis may require the identification of the interactions among system elements. These interactions lead to the SE process of interface analysis. Integral to this aspect is the principle of interactions. Interactions occur both with other system elements as well as with external elements and the environment. In a systems approach, interfaces have both a technical and managerial importance. Managerial aspects include the contracts between interfacing organizations. Technical aspects include the properties of the physical and functional interfaces. Browning provides a list of desirable characteristics of both technical and managerial interface characteristics (Browning 2009, 1418-1419) .

System synthesis will include activities to understand the properties of system elements, the structure of proposed system solutions, and the resultant behavior of the composed system. A number of system concepts for describing system behavior are discussed in Concepts of Systems Thinking topic. It should be noted that in order to fully understand a system's behavior, we must consider the full range of environments in which it might be placed and its allowable state in each. According to Page, in complex systems, the individual elements of the system are characterized by properties which enhance the systems as a whole, such as their adaptability (Page 2009).

Defining the System-of-Interest

Flood and Carson provide two ways to identify system boundaries: a bottom-up, or **structural approach**, which starts with significant system elements and builds out, and a top down, or **behavioral approach**, in which major systems needed to fulfill a goal are identified and then the work flows downward (Flood and Carson 1993). They identify a number of rules proposed by Beishon (Beishon 1980) and Jones (Jones 1982) to help in the selection of the best approach.

In either case, the ways in which system elements are refined, grouped, and allocated must be driven towards the synthesis of a realizable system solution description. A realizable solution must consider elements that are either already available, can be created from existing system elements, or are themselves described as system contexts which will need to be synthesized at a future point. In the third case, it is one of the outcomes of the Analysis and Selection between Alternative Solutions activities that is used to assess the risk that a given element may not be able to be synthesized in the required time limit or cost budget.

A top down approach might start with a system boundary and an overall description of system functions. Through the repeated application of element identification, division, grouping, and allocation of functions, a complete description of the elements needed for the SoI can be defined. In this case, the choice of system elements and allocation of functions may be guided by pre-defined ways of solving a given problem or by identified system patterns; both can support as well as insert bias into the synthesis. For example, one might start with the need to provide energy to a new housing project and propose solution options based around connections to an existing power grid, local power generators, renewable energy sources, increased energy efficiency, etc.

The iterative nature of analysis also reflects the need to change the solution as the life cycle progresses and changes the system's environment; thereby, possibly changing what a "best" solution is.

A bottom up approach starts with major elements and interactions. Again, division, grouping, and identification allows for the construction of a full system description that is capable of providing all the necessary functions, at which point the final SoI boundary can be set. In this case, the choice of system elements and groupings will be driven by the goal of ensuring that the major system elements can be formed together into a viable system whole. For example, there may be a need to replace an existing delivery vehicle and produce solution options that consider vehicle ownership/leasing, driver training, petrol, diesel or electric fuel, etc.

The systems approach aspect of synthesis leads to SE terms such as “design” and “development.” Wasson describes synthesis from a SE point of view (Wasson 2006, 390-690). White provides a comprehensive discussion of methods of achieving design synthesis (White 2009, 512-515). The systems approach treats synthesis at the abstract level while the SE process definitions provide the concrete steps.

The SoI brings together elements, sub-systems and systems through the concept of synthesis to identify a solution option.

Synthesis of possible solutions may result in the development of artifacts documenting the synthesis itself and provide the basis for analysis and selection between alternative solutions. These artifacts are dynamic and will change as the SoI changes its environment throughout the system life cycle.

References

Works Cited

Beishon, J. 1980. *Systems Organisations: the Management of Complexity*. Milton Keynes; Open University press.

Blanchard, B. and W.J. Fabrcky. 2006. *Systems Engineering and Analysis*. Upper Saddle River, NJ: Prentice Hall.

Boehm, B. 2009. "Applying the Incremental Commitment Model to Brownfield System Development". Proceedings of the 7th Annual Conference on Systems Engineering Research (CSER), Loughborough, UK.

Browning, T.R. 2009. "Using the Design Structure Matrix to Design Program Organizations". In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Buede, D.M. 2009. "Functional Analysis". In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Checkand, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Flood, R.L. and E.R. Carson. 1993. *Dealing with Complexity: An Introduction to the Theory and Application of Systems Science*, 2nd ed. New York, NY, USA: Plenum Press.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight*. 12(4) (December 2009): 59-63.

INCOSE. 1998. "INCOSE SE Terms Glossary." INCOSE Concepts and Terms WG (eds.). Seattle, WA, USA: International Council on Systems Engineering.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?". INCOSE *Insight*. 13(1) (April 2010): 41-43.

Jones, L. 1982. "Defining System Boundaries in Practice: Some Proposals and Guidelines." *Journal of Applied Systems Analysis*, 9: 41-55.

Levin, A.H. 2009. "System Architectures". In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Page, S.E. 2009. "Understanding Complexity". The Great Courses. Chantilly, VA, USA: The Teaching Company.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

White, Jr., K.P. 2009. "Systems Design." In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Primary References

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4) (December 2009): 59-63.

ISO/IEC/IEEE. 2015. Systems and software engineering -- System life cycle processes. Geneva, Switzerland: International Organisation for Standardisation/International Electrotechnical Commissions / Institute of Electrical and Electronics Engineer. ISO/IEC/IEEE 15288:2015.

Jackson, S., D. Hitchins and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight*. 13(1) (April 2010): 41-43.

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Analysis and Selection between Alternative Solutions

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the analysis and selection of a preferred solution from the possible options, which may have been proposed by Synthesizing Possible Solutions. Selected solution options may form the starting point for Implementing and Proving a Solution. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

System Analysis

System analysis is an activity in the systems approach that evaluates one or more system artifacts created during the activities involved in Synthesizing Possible Solutions, such as:

- Defining assessment criteria based on the required properties and behavior of an identified problem or opportunity system situation.
- Accessing the properties and behavior of each candidate solution in comparison to the criteria.
- Comparing the assessments of the candidate solutions and identification of any that could resolve the problem or exploit the opportunities, along with the selection of candidates that should be further explored.

As discussed in Synthesizing Possible Solutions topic, the problem context for an engineered system will include a logical or ideal system solution description. It is assumed that the solution that “best” matches the ideal one will be the most acceptable solution to the stakeholders. Note, as discussed below, the “best” solution should include an understanding of cost and risk, as well as effectiveness. The problem context may include a soft system conceptual model describing the logical elements of a system to resolve the problem situation and how these are perceived by different stakeholders (Checkland 1999). This soft context view will provide additional criteria for the analysis process, which may become the critical issue in selecting between two equally effective solution alternatives.

Hence, analysis is often not a one-time process of solution selection; rather, it is used in combination with problem understanding and solution synthesis to progress towards a more complete understanding of problems and solutions over time (see Applying the Systems Approach topic for a more complete discussion of the dynamics of this aspect of the approach).

Effectiveness Analysis

Effectiveness studies use the problem or opportunity system context as a starting point.

The effectiveness of a synthesized system solution will include performance criteria associated with both the system's primary and enabling functions. These are derived from the system's purpose, in order to enable the realization of stakeholder needs in one or more, wider system contexts.

For a product system there are a set of generic non-functional qualities that are associated with different types of solution patterns or technology, e.g., safety, security, reliability, maintainability, usability, etc. These criteria are often explicitly stated as parts of the domain knowledge of related technical disciplines in technology domains.

For a service system or enterprise system the criteria will be more directly linked to the identified user needs or enterprise goals. Typical qualities for such systems include agility, resilience, flexibility, upgradeability, etc.

In addition to assessments of the absolute effectiveness of a given solution system, systems engineers must also be able to combine effectiveness with the limitations of cost and timescales included in the problem context. In general, the role of system analysis is to identify the proposed solutions which can provide some effectiveness within the cost and time allocated to any given iteration of the systems approach (see Applying the Systems Approach for details). If none of the solutions can deliver an effectiveness level that justifies the proposed investment, then it is necessary to return to the original framing of the problem. If at least one solution is assessed as sufficiently effective, then a choice between solutions can be proposed.

Trade-Off Studies

In the context of the definition of a system, a trade-off study consists of comparing the characteristics of each candidate system element to those of each candidate system architecture, in order to determine the solution that globally balances the assessment criteria in the best way. The various characteristics analyzed are gathered in cost analysis, technical risks analysis, and effectiveness analysis (NASA 2007). To accomplish a trade off study there are a variety of methods, often supported by tooling. Each class of analysis is the subject of the following topics:

- Assessment criteria are used to classify the various candidate solutions. They are either absolute or relative. For example, the maximum cost per unit produced is c$, cost reduction shall be x%, effectiveness improvement is y%, and risk mitigation is z%.
- **Boundaries** identify and limit the characteristics or criteria to be taken into account at the time of analysis (e.g., the kind of costs to be taken into account, acceptable technical risks, and the type and level of effectiveness).
- **Scales** are used to quantify the characteristics, properties, and/or criteria and to make comparisons. Their definition requires knowledge of the highest and lowest limits, as well as the type of evolution of the characteristic (linear, logarithmic, etc.).
- An assessment score is assigned to a characteristic or criterion for each candidate solution. The goal of the trade-off study is to succeed in quantifying the three variables (and their decomposition in sub-variables) of cost, risk, and effectiveness for each candidate solution. This operation is generally complex and requires the use of models.
- The **optimization** of the characteristics or properties improves the scoring of interesting solutions.

A decision-making process is not an accurate science; ergo, trade-off studies have limits. The following concerns should be taken into account:

- Subjective Criteria – personal bias of the analyst; for example, if the component has to be beautiful, what constitutes a "beautiful" component?
- Uncertain Data – for example, inflation has to be taken into account to estimate the cost of maintenance during the complete life cycle of a system, how can a systems engineer predict the evolution of inflation over the next five years?

- Sensitivity Analysis – A global assessment score that is designated to every candidate solution is not absolute; thus, it is recommended that a robust selection is gathered by performing a sensitivity analysis that considers small variations of assessment criteria values (weights). The selection is robust if the variations do not change the order of scores.

A thorough trade-off study specifies the assumptions, variables, and confidence intervals of the results.

Systems Principles of System Analysis

From the discussions above, the following general principles of systems analysis can be defined:

- Systems analysis is an iterative activity consisting of trade studies made between various solution options from the systems synthesis activity.
- Systems analysis uses assessment criteria based upon a problem or opportunity system description.
  - These criteria will be based around an ideal system description that assumes a hard system problem context can be defined.
  - The criteria must consider required system behavior and properties of the complete solution in all of the possible wider system contexts and environments.
  - Trade studies require equal consideration to the primary system and the enabling system working as a single sytem to address the User need. These trades need to consider system requirements for Key Performance Parameters (KPPs), systems safety, security, and affordability across the entire life cycle
  - This ideal system description may be supported by soft system descriptions from which additional "soft" criteria may be defined (e.g., a stakeholder preference for or against certain kinds of solutions and relevant social, political, or cultural conventions to be considered in the likely solution environment, etc.).
- At a minimum, the assessment criteria should include the constraints on cost and time scales acceptable to stakeholders.
- Trade studies provide a mechanism for conducting analysis of alternative solutions.
  - A trade study should consider a "system of assessment criteria", designating special attention to the limitations and dependencies between individual criteria.
  - Trade studies need to deal with both objective and subjective criteria. Care must be taken to assess the sensitivity of the overall assessment to particular criteria.

References

Works Cited

Checkland, P.B. 1999. *Systems Thinking, Systems Practice*. Chichester, UK: John Wiley & Sons Ltd.

NASA. 2007. *Systems Engineering Handbook*, Revision 1. Washington, DC, USA: National Aeronautics and Space Administration (NASA). NASA/SP-2007-6105.

Primary References

ISO/IEC/IEEE. 2015. *Systems and software engineering -- System life cycle processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / / Institute of Electrical and Electronics Engineer. ISO/IEC/IEEE 15288:2015.

Jackson, S., D. Hitchins and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1) (April 2010): 41-43.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Implementing and Proving a Solution

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the implementation and proving of a preferred solution that may have been selected by activities described in the Analysis and Selection between Alternative Solutions topic. The activities that apply to an implemented solution during its operational life are described in Deploying, Using, and Sustaining Systems to Solve Problems topic and how systems fit into commercial and acquisition relationships is discussed in the Stakeholder Responsibility topic. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Proving the System Overview

This topic covers both the sub-topics of verification and validation.

Verification

Verification is the determination that each element of the system meets the requirements of a documented specification (see principle of elements). Verification is performed at each level of the system hierarchy. In the systems approach this topic pertains to the more abstract level of providing evidence that the system will accomplish what it was meant to do. In SE this topic pertains to providing quantitative evidence from tests and other methods for verifying the performance of the system.

Validation

Validation is the determination that the entire system meets the needs of the stakeholders. Validation only occurs at the top level of the system hierarchy. In the systems approach this topic pertains to the more abstract level of making sure the system meets the needs of the stakeholders. In SE, this topic pertains to the detailed demonstrations and other methods that are used to promote stakeholder satisfaction.

In a SE context, Wasson provides a comprehensive guide to the methods of both system verification and system validation (Wasson 2006, 691-709).

References

Works Cited

Wasson, C. S. 2006. *System Analysis, Design, and Development.* Hoboken, NJ, USA: John Wiley & Sons.

Primary References

Jackson, S., D. Hitchins and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight*. 13(1) (April 2010): 41-43.

Additional References

MITRE. 2012. "Verification and Validation." *Systems Engineering Guide.* http:/ / mitre. org/ work/ systems_engineering/ guide/ se_lifecycle_building_blocks/ test_evaluation/ verification_validation. html (accessed September 11, 2012)

Wasson, C. S. 2006. *System Analysis, Design, and Development.* Hoboken, NJ, USA: John Wiley & Sons.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Deploying, Using, and Sustaining Systems to Solve Problems

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the deployment, sustainment, and use of a solution, that may have been developed through the activities described in the Implementing and Proving a Solution topic. Discussion of how a deployed systems fit into commercial and acquisition relationships in Stakeholder Responsibility topic. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Introduction

Part 3, Systems Engineering and Management, of the Guide to the SE Body of Knowledge (SEBoK) provides two additional KAs that address the engineering aspects of these steps of the systems approach. In KA Product and Service Life Management and System Deployment and Use, in Part 3 explain the SE aspects of deployment, operation, maintenance, logistics, service life extension, updates, upgrades, disposal and the retirement of systems.

A systems approach considers the total system and the total life cycle of the system. This includes all aspects of the system and the system throughout its life until the day users depose of the system and the external enterprises complete the handling of the disposed system products. Creation of the system is rarely the step that solves the stakeholders' problems. It is the use of the system solution that solves the problem. So from this perspective the deployment, use and sustainment of the system are important concepts that must be a part of the systems approach.

Engineered systems are eventually owned by an individual, team, or enterprise. Those who own the system during development may not be the ones who own the system when it is in operation. Moreover, the owners may not be the

users; e.g., service systems may be used by the general public but owned by a specific business that is offering the service. The transition of a system from development to operations is often itself a complex task, involving such activities as training those who will operate the system, taking legal actions to complete the transfer, and establishing logistical arrangements so that the operators can keep the system running once the transition is completed.

A complete systems approach must also consider the many enterprises involved in the system from initial conception through the completion of the disposal process. These enterprises are all stakeholders with requirements , which all have interfaces that must be considered as part of a total systems approach.

There is very little in the literature pertaining to the application of the systems approach to these phases of the life cycle. However, a basic premise of this KA is that the systems approach pertains to all phases of a system's life cycle. Hence, to properly build systems to solve problems or for other uses, it can be inferred that the systems approach pertains to the deployment, use, and the sustainment of the systems. Many of the available references in this topic area are from SE literature rather than from literature associated with the systems approach so the reader should also see Part 3, Systems Engineering and Management, of the SEBoK.

Deployment: The Transition from Development to Operation

Transferring custody of the SoI and responsibility for its support from one organization to another occurs during deployment and is often called transition (INCOSE 2011). Transition of a product system includes the integration of the system into the acquiring organization's infrastructure. Deployment and transition involves the activity of moving the system from the development to the operational locations, along with the support systems necessary to accomplish the relocation.

Transition includes the initial installation of a system and the determination that it is compatible with the wider system and does not cause any significant wider system issues. This process of acceptance and release for use varies between domains and across businesses and enterprises and can be thought of as an initial assessment of the system's effectiveness (Hitchins 2007). Generally, transition may be considered as having two parts: 1.) ensuring that the new system interoperates with the systems around it and 2.) ensuring the resulting system is safe and possesses other critical operational properties.

It is particularly important to considered emergent properties when a new system is added to the existing organization's system of systems (SoS) network, as well as the complexity of the organization into which the new system is transitioned (see also Complexity). The more complex the receiving organization is the more challenging the transition will be, and the greater the likelihood of unintended interactions and consequences from the new system's insertion. Dealing with the consequences of this complexity starts in transition and continues into operation, maintenance, and disposal.

Transition of a service system is often performed in two stages. First, the service system infrastructure is accepted and released. Second, each realization of the service is accepted and released. There can be significant problems during the second stage if the required responsiveness of the service does not leave sufficient time to ensure that the service meets necessary functional and quality attributes, including interoperability, safety, and security. (See Service Systems Engineering).

Transition and deployment of a system may introduce unique requirements that are not necessary for operation or use. These requirements can influence the design of the system; therefore, must be considered during the initial requirements and design stages. The most common examples are related to the need to transport the system or system elements, which often limits the size and weight of the system elements.

Transition can also require its own enabling systems, each of which can be realized using a systems approach.

Use: Operation

Use of the system to help enable delivery of user services is often called "operations" (INCOSE 2011). A system's effectiveness is normally considered throughout the operational life of a system. For a complex system, emergent behavior should be considered in three ways:

- to identify and plan for emergent properties within the system realization process (See System Realization KA in Part 3, Systems Engineering and Management)
- to incorporate mechanisms for identifying and handling unexpected emergent properties within the system during its use
- to provide necessary procedures for dealing with wider system consequences of unexpected emergent properties in the enterprise (e.g., emergency responses or medical first aid)

Operations require their own enabling systems, each of which can be realized using a systems approach.

System Sustainment and Maintenance

System sustainment requires maintenance of the system throughout its useful life (INCOSE 2011). In system terms, maintenance implements systems that handle entropy and maintaining the SoI in a viable state. Since an open system maintains its existence by continual exchange of energy, information, and materiel with it's environment, one aspect of its maintenance must be the management of resources in the environment.

Hitchins (Hitchins 2007) describes generic approaches to resource management and viability management based on systems concepts. Resource management identifies the need to consider the acquisition, storage, distribution, conversion, and disposal of resources. Viability management should consider systems to maintain homeostasis and a means for ensuring resilience to environmental disturbance and adaptability to environmental change.

Maintenance will require its own enabling systems, each of which can be realized using a systems approach. Maintenance success is more likely if it is considered as part of the system concept and design well before the system enters service.

Disposal

A total life cycle systems approach cannot be considered complete without consideration of how disposal of the system will be accomplished. The purpose of disposal is to remove a system element from the operational environment with the intent of permanently terminating its use and remove any hazardous or toxic materials or waste products (INCOSE 2011).

During disposal the entirety of the open system crosses the boundary from the system side to the environment. A complete systems approach must consider how it crosses the boundary and what remains that must be managed by enterprises other than the ones that developed, used or sustained the system. Including disposal in the system approach expands the stakeholders, the enterprises and the external systems that must be considered.

Disposal requires its own enabling systems, each of which can be realized using a systems approach. Some of these may be contained within the system boundaries and others may be external to the system. For the external disposal systems, the interface where the handover occurs must be considered. As with maintenance, a large part of successful disposal requires related issues to have been considered early on in the system's life cycle.

The topic Disposal and Retirement in Part 3, Systems Engineering and Management, of the SEBoK provides information on the engineering aspects of system disposal.

References

Works Cited

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* Hoboken, NJ, USA: John Wiley and Sons.

INCOSE. 2012. *INCOSE Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TP-2003-002-03.2.2.

Primary References

INCOSE. 2011. *Systems Engineering Handbook*, version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TP-2003-002-03.2.1.

Additional References

MITRE. 2011. "Transformation Planning and Organizational Change." . *Systems Engineering Guide.* Accessed December 4 2014 at MITRE http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1., released 16 October 2018

Stakeholder Responsibility

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It summarizes various aspects of stakeholder responsibility for acquisition and ownership during the system life cycle processes covered by such sources as the International Council on Systems Engineering Handbook (INCOSE 2012). Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Products, Services, and Enterprises

Most often, the terms "product" and "service" describe the effects that are exchanged in a customer and supplier agreement. This may be a commercial agreement, one funded publicly by a charity, or provided by a government agency. The difference between a product and a service is that a product is an artifact acquired to achieve an outcome while a service is an outcome supplied directly to a user.

The terms “customer” and “user” are often used interchangeably in engineering and management disciplines. The INCOSE *Systems Engineering Handbook* (INCOSE 2012) makes the following specific distinctions among the stakeholders associated with a system:

- The acquirer is the stakeholder that acquires or procures a product or service from a supplier.
- The supplier is an organization or individual that enters into an agreement with the acquirer to supply a product or service.

- The operator is an individual or organization that that uses knowledge, skills and procedures to perform the functions of the system to provide the product or service.
- The user or customer is the individual or group that benefit from the operation of the system.

These terms define the roles stakeholders take; however, they may not always lie within these distinct entities (e.g. sometimes the acquirer may also be the user). This also applies to service systems, as some of the entities may also overlap in roles. Parnell et al., (Parnell et al. 2011) offer an alternative list of stakeholders that include decision authority, client, owner, user, consumer, and interconnected.

Product systems consist of hardware, software, and humans, and they have traditionally been the focus of SE efforts. These systems are delivered to the acquirer and operated to accomplish the goals that led to the requirements for the system. These requirements were derived from the need to provide products and services to one or more users as part of an enterprise.

The delivery (supplying) of a service is indicative of the direct delivery of an outcome, which is often related to the delivery of products (e.g., a maintenance, training, or cleaning service). This is not the same as the delivery of a service system (see the discussion below).

In traditional SE, the term “service” or “service system” refers to the wider system context that describes the acquirer's need to deliver user value. In this case, the service system is a fixed system definition that dictates the manner in which the acquiring enterprise will utilize the products to enable the delivery of services to users. Product systems are designed to be integrated and operated as appropriate to enable this service to be maintained or improved as required. In this view, a service system is static and contains dedicated products, people, and resources; that is, hierarchies of products are engineered to provide acquirers with the ability to offer predefined services to users or customers.

More recently, the term "service systems" has been used to describe a system that is engineered in a manner that allows enterprises to offer services directly to users, bypassing the need to hold all of the necessary products and services within the enterprise itself. This requires the expansion of the definition of a “supplier” as follows:

- A **product supplier** is an organization or individual that enters into an agreement with an acquirer to supply a product or related product support services.
- A **service system supplier** is an organization or individual that enters into an agreement with an acquirer to supply a service system.
- A **service supplier** is an organization or individual that enters into an agreement with a user to supply a service.

These service systems tend to be configured dynamically to deal with problems that traditional static service find challenging to address. This view of a service system employs "late binding" with product systems that are not owned by the enterprise, but are used to enable the service to be offered as closely to given time demands as possible. This is the definition of a service system used in the Service Systems Engineering topic in Part 4, Applications of Systems Engineering.

Stakeholder Needs

One of the most critical stakeholder responsibilities is to identify the needs and requirements for the system that provides the products or services (INCOSE 2012).These needs and requirements are expressed in agreements between acquirers and suppliers.

There are other stakeholders who shape system requirements based on their needs, but who are not necessarily acquirers or suppliers. The stakeholders and the requirements engineers share the responsibility to identify their needs during the requirements process.

Acquirer/Supplier Agreements

Lawson (Lawson 2010) provides a perspective on what it means to own systems, trade in system products and services, and the implications of supply chains in respect to the value added and ownership of the systems, its products and services. INCOSE (INCOSE 2012) defines two life cycle processes related to acquisition and supply. The acquisition process includes activities to identify, select, and reach commercial agreements with a product or service supplier.

In many larger organizations, there is a tradition of system ownership vested in individuals or, in some cases, enterprise entities (groups or teams). Ownership implies the authority and responsibility to create, manage, and dispose of a system-of-interest (SoI), as well as sometimes to operate the SoI.

Product Acquire/Supply

In some industries, a supplier works directly with an acquirer to help understand the acquirer's needs and then engineer one or more products to satisfy these needs. In certain cases, a single supplier will provide the complete worthy product system. In other cases, a supply chain will be formed to deliver product systems with a system integrator to ensure they fit together and integrate into the wider context. This is a theoretical view of product systems engineering in which the context is fixed and the product is designed to fit into it. A good systems engineer may suggest changes to the enterprise as a better way to solve the problem and then modify the product system's requirements accordingly. However, at some point, an agreed context will be set and a product system developed to work within it.

For many commercial products, such as mobile phones, a supplier creates a representative user profile to generate the requirement and then markets the product to real users once it is realized. In these cases, the other elements of the systems approach are performed by the acquirer/user and may not follow formal SE processes. It is important that a product supplier takes this into account when considering the best manner in which to engineer a system, as additional help or support services may need to be offered with the purchased product. The idea of a supplier offering support services for users with a type of product purchased elsewhere (e.g., an auto-mechanic servicing different makes of cars) begins to overlap with the service systems context, as discussed in the next topic.

For an institutionalized infrastructure in which SoIs are entirely owned by an enterprise or parties thereof, the entire responsibility of life cycle management, including operation, is often vested with the system owners. These systems belong to the system asset portfolio of an enterprise or multiple enterprises and provide the system resources, including the planned systems that are developed during life cycle management.

Service Acquire/Supply

Organizations providing service systems need not own the individual products and services that they deliver to their users and customers. With this viewpoint, the supplied service system includes the means to identify and gain access to appropriate products or services when needed. The service systems then would then be the bundle of products and services assembled for the user; for example, assembling software applications and service agreements for a mobile phone already owned by a user. The enterprises providing service systems may, in turn, offer infrastructure services to a wide range of different technologies or application domains. This can mean that the transition, operation, maintenance and disposal activities associated with system ownership may not be embedded in the acquiring service system enterprise, and will therefore need to be treated as separate system services. More detail can be found in Product Systems Engineering, Service Systems Engineering, and Enterprise Systems Engineering, in Part 4, Applications of Systems Engineering in the *Guide to the Systems Engineering Body of Knowledge* (SEBoK).

The service systems engineer helps the service supplier create and sustain the service system that can be used to discover, integrate, and use specific versions of generic products or services when needed. The realization of service systems requires the ability to make use of product systems; however, these product systems are developed and

owned outside of the service system. The service system must be able to gain access to a product or service when needed, as well as to interface with it effectively. The use of open interface standards, such as standard power supplies, interface connections (e.g., Universal Serial Bus (USB)), or file formats (e.g., Portable Document Format (PDF)) can help make this easier.

Enterprise Evolution

A useful distinction between product system design and enterprise system design is that "enterprise design does not occur at a single point in time like the design of most systems. Instead, enterprises evolve over time and are constantly changing, or are constantly being designed" (Giachetti 2010, xiii).

The enterprise developer may also aim to optimize back stage processes (the internal operations) of an organization or an institution by exploiting advances in technology, particularly information technology (IT) and associated processes. In these cases, the engineered systems are considered to be enterprise systems.

Enterprise systems may offer products (goods) and/or services. From an enterprise engineering viewpoint, an enterprise concurrent with its product SE must not only look at the development and delivery of the products but also look at the alignment and optimization of the product delivery within the enterprise objectives. Similarly, in service SE, the main focus is on an intangible value delivery to the end-customer (externally focused: front stage), in which internal and external processes must be synchronized. However, with the rapid advances in information and communications technologies (ICT), in many cases the boundaries between internal and external processes are quite blurred. Current SE research is extending product methods, processes, and tools into the enterprise transformation and service innovation fields to exploit advances in business process methodologies and technologies.

Enterprise SE must not only do the engineering of the enterprise itself, but may also be involved in the engineering of the service systems and products systems that are necessary for the enterprise to achieve its goals.

References

Works Cited

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods.* Boca Raton, FL, USA: CRC Press.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Parnell, G.S., P.J. Driscoll, and D.L Henderson (eds). 2011. *Decision Making for Systems Engineering and Management*, 2nd ed. Wiley Series in Systems Engineering. Hoboken, NJ: Wiley & Sons Inc.

Primary References

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.Systems and Software Engineering -- System Life Cycle Processes. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Additional References

Carlock, P.G. and R.E. Fenton. 2001. "System of Systems (SoS) enterprise systems engineering for information-intensive organizations." *Systems Engineering.* 4(4): 242–261.

Rouse, W.B. 2005. "Enterprises as Systems: Essential Challenges and Approaches to Transformation." *Systems Engineering.* 8(2): 138-150.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Applying the Systems Approach

The systems approach relates to both the dynamics of problem resolution and stakeholder value over time, as well as to the levels of system relationship, detailed management, and the engineering activities this implies.

This article builds on the concepts introduced in Overview of the Systems Approach topic. It is part of the Systems Approach Applied to Engineered Systems knowledge area (KA), which describes, primarily through five groups of activities, the application of an approach based around systems thinking to engineered system contexts throughout their lives.

Life Cycle

Engineered Systems provide outcomes which deliver benefits to stakeholders by helping them achieve something of value in one or more problem situations. Ultimately, a system is successful only if it enables successful outcomes for its stakeholders (Boehm and Jain 2006). In complex real world situations value can best be provided through a continuing process of adapting the system needs and developing associated solutions in response to changing circumstances, according to the principle of **Progressive Satisfying** (Hitchins 2009).

A value cycle associating the systems approach to the deliver real world stakeholder benefits is discussed in the Overview of the Systems Approach topic. A greater understanding of the value of an engineered system within its context enables agreement on the problem situation and appropriate system interventions to be created, deployed, and used overall, in turn enabling a more effective application of the systems approach. Value is fully realized only when considered within the context of time, cost, funding and other resource issues appropriate to key stakeholders (Ring 1998).

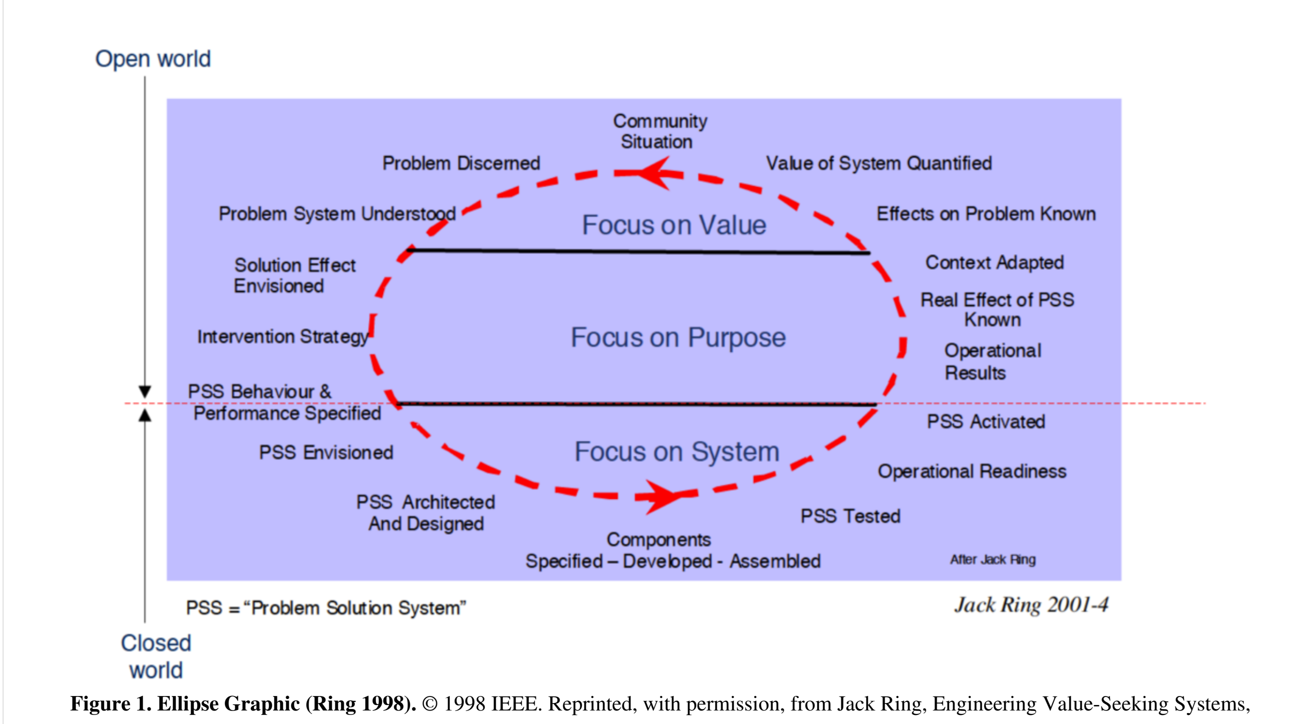


Figure 1. Ellipse Graphic (Ring 1998). © 1998 IEEE. Reprinted, with permission, from Jack Ring, Engineering Value-Seeking Systems, IEEE-SMC. Conference Proceedings.

The views in Figure 1 apply the idea of **Systemic Intervention** to the resolution of problem situations in which one or more engineered system solution might be required. For each turn of the cycle an agreement is made between stakeholders and developers that an Engineered System to solve problem X with effectiveness Y in agreed conditions Z has a chance of delivering value A for which we are willing to invest cost B and other resources C.

It is in the nature of wicked problems that this proposition cannot be a certainty. life cycle approaches to understand and manage the shared risk of tackling such problems are discussed in Life Cycle Models. The idea of Systemic Intervention comes from soft systems thinking (see Systems Approaches).

For each of the engineered system problems, the solutions agreed above must be developed such that they are effective in terms of cost, performance and other properties relevant to the problem domain. A developer must consider not only what to do, but when and how much to do to provide real value (Senge 1990). In systems engineering (SE) and management practices this leads to the two key concepts (INCOSE 2011):

- **Life Cycles:** Stakeholder value and problem resolution described as a set of life cycle stages over which problems can be explored and resolved, and resources can be managed.
- **Life Cycle Processes:** Systems of activities focused on creation and sharing of knowledge associated with the systems approach, that can be employed to promote a holistic approach over a life cycle.

Life cycle management provides the framework in which to take a systems approach to all aspects of an engineered system context, which includes not only the system product or service but also the systems to create, deploy and support it (Martin 2004). The following sections consider how the systems approach should be applied to an identified problem statement, within the context of the overall value cycle discussed above.

Application Principles

Concurrency

Within any application of the systems approach the activities of problem identification, solution synthesis and selection; solution implementation and proving and deployment, sustainment and use should be applied concurrently, reflecting their interrelationships and dependencies.

The system value cycle (Ring 1998) can be taken as a generic model of the life of an engineered system within a problem resolution cycle driven by stakeholder value. For practical reasons it is necessary to break this life down into a set of finite stages, to allow activities to be organized. We can express the value cycle as six groups of questions to cycle around value, problem, and solution questions that are related to the systems approach:

1. What values do Stakeholders want/need?
2. What system outcomes could improve this value?
3. What system can provide these outcomes?
4. How do we create such a system?
5. How do we deploy and use the system to achieve the outcomes?
6. Do these outcomes provide the expected improvement in value?

The above questions focus on each iteration of the systems approach to deliver stakeholder goals within an enterprise context. Activities 1 and 6 are part of the business cycles of providing stakeholder value within an enterprise, whereas activities 2 through 5 can be mapped directly to product, service, and enterprise engineering life cycles. A distinction is made here between the normal business of an enterprise and the longer-term strategic activities of Enterprise Systems Engineering.

The following diagram illustrates the concurrent nature of the activities of a systems approach over time.

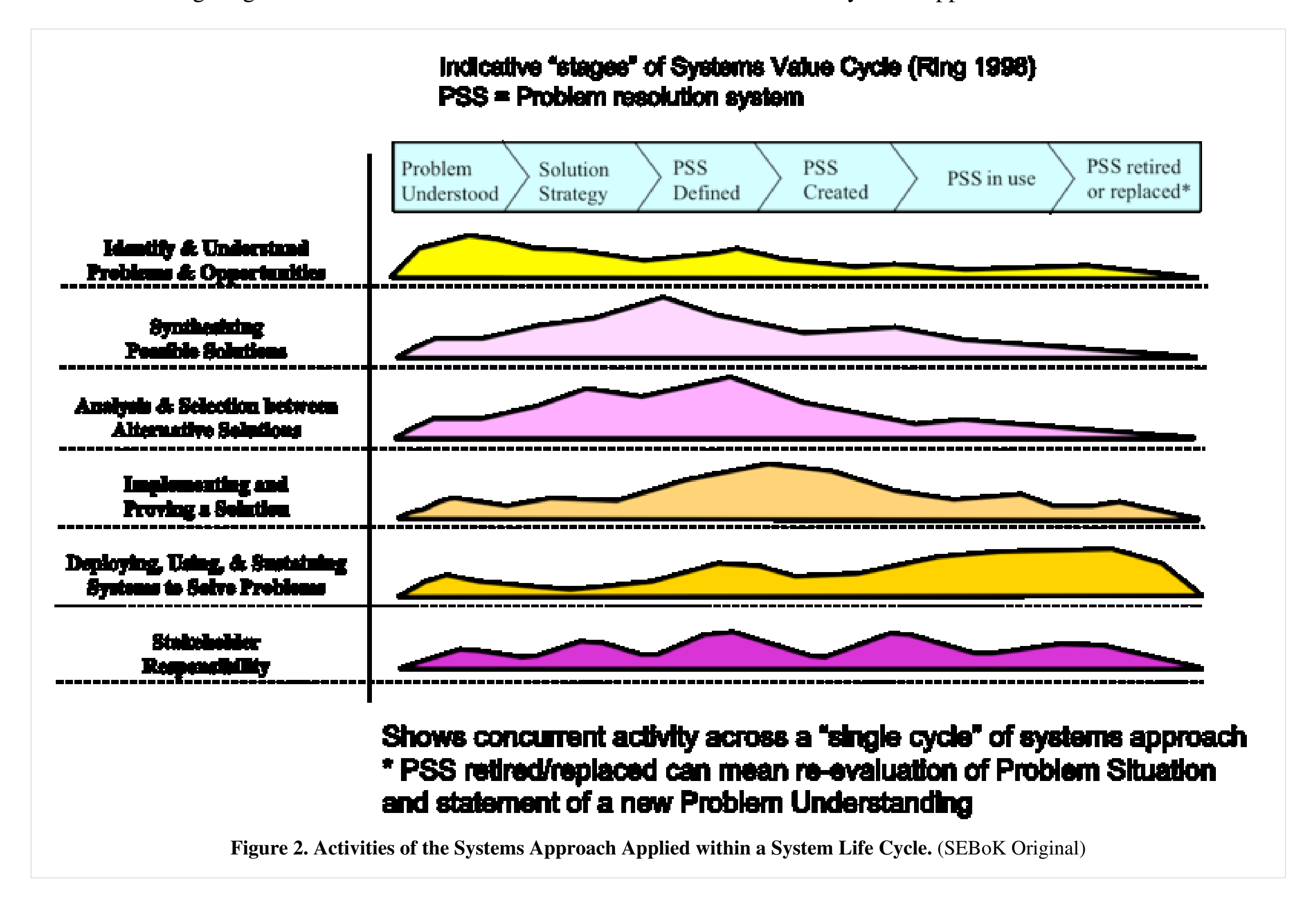


Figure 2. Activities of the Systems Approach Applied within a System Life Cycle. (SEBoK Original)

The lines on Figure 2 represent activity in each of the activity areas over a simple (not to scale) life cycle based on the questions above. Activities may have a primary focus in certain stages, but need to span the whole of life to ensure a holistic approach. For example, problem identification has a large input during the Problem Understanding stage, but problems are refined, reviewed, and reassessed over the rest of the life cycle. Similarly, Implement and Proving activities are conducted during the transition from Create to Use. This is only possible if proving issues, strategies, and risks are considered in earlier stages. This diagram is a schematic representation of these activity mappings, sometimes called a hump diagram (Kruchten 2003).

For the generic systems approach, the following fundamental life cycle principles apply:

- A life cycle has groups of stages which cover understanding stakeholder value; exploration of a problem situation (see System Definition); creation of a system solution (see System Realization); and System Deployment and Use.
- Life cycle processes define a system of engineering and management activities based on the detailed information needed to ensure a systems approach across a life cycle (e.g., requirements, architecture, verification, and validation).
- Activities in any of the processes may be employed in all of the stages to allow for appropriate concurrency.
- The sequence and control of the life cycle stages and concurrent process activities must be tailored to the problem situation and commercial environment (Lawson 2010), thus leading to the selection of an appropriate life cycle model.
- Appropriate management activities must be included in the life cycle to ensure consideration of time, cost, and resource drivers.
- In focusing on the creation of a specific system-of-interest (SoI) to provide solutions within the cycle, it is important to recognize the need to employ the right balance between reductionism and holism by considering the appropriate system context.

The ways in which this idea of concurrent process activity across a life cycle has been implemented in SE are discussed in Systems Engineering and Management.

Iteration

The systems approach can be applied in an iterative way to move towards an acceptable solution to a problem situation within a larger cycle of stakeholder value.

The systems approach can be applied to multiple systems within an engineered system context, as discussed below. At each level, the approach may be applied iteratively to cycle between what is needed and versions of the solutions within a life cycle model.

Hitchins (Hitchins 2009) defines two principle related to iterations:

- **Adaptive Optimizing —** Continual redesign addresses the problem space, detecting and addressing changes in situation, operational environment, other interacting systems, and other factors; it continually conceives, designs, and implements or reconfigures the whole solution system to perform with optimal effectiveness in the contemporary operational environment.
- **Progressive Entropy Reduction —** Continual performance and capability improvement of systems in operation may be undertaken by customer or user organizations with or without support from industry, as they seek to “get the best” out of their systems in demanding situations. In terms of knowledge or information, this process involves progressively reducing entropy, going from a condition of high entropy (that is, disorder) at the outset to low entropy (order) at the finish.

In general, these two cycles of iterations can be realized from combinations of three life cycle types (Adcock 2005):

- Sequential: With iteration between the stages to solve detailed issues as they arise, a single application of the systems approach is sufficient.

- Incremental: Successive versions of the sequential approach are necessary for a solution concept. Each increment adds functionality or effectiveness to the growing solution over time.
- Evolutionary: A series of applications of the sequential approach for alternative solutions intended to both provide stakeholder value and increase problem understanding. Each evolutionary cycle provides an opportunity to examine how the solution is used so these lessons learned can be incorporated in the next iteration.

These aspects of the systems approach form the basis for life cycle models in Life Cycle Models.

Recursion

The stakeholder value, problem resolution, and system creation aspects of the system value cycle may each require the use of a focused systems approach. These might be soft systems to prove a better understanding of a situation, product systems and/or service systems solutions to operational needs, enabling systems to support an aspect of the product or service life cycle, or enabling systems used directly by the enterprise system.

Each of these systems may be identified as a system-of-interest (SoI)and require the application of the systems approach. This application may be sequential (the start of one system approach dependent on the completion of another) or parallel (independent approaches which may or may not overlap in time), but will often be recursive in nature.

Recursion is a technique borrowed from computer science. In computer science recursion occurs when a function calls itself repeatedly to logically simplify an algorithm. In a recursive application applied to systems, the systems approach for one system-of-interest is nested inside another. Examples include cases where

- trades made at one level of the system require trades to be made for system elements;
- the analysis of a system requires analysis of a system element;
- the synthesis of a solution system requires one or more sub-system elements; and
- the verification of a product system requires verification of system elements.

In each case, the “outer” system approach may continue in parallel with the “inner” to some extent, but depends on key outcomes for its own progress.

As with all recursive processes, at some stage the application of the approach must reach a level at which it can be completed successfully. This then "rolls up" to allow higher levels to move forward and eventually complete all nested applications successfully.

The INCOSE *Systems Engineering Handbook* (INCOSE 2011) describes a recursive application of SE to levels of system element with each application representing a system project. Martin (1997) describes the recursive application of SE within a product system hierarchy until a component level is reached, at which point procurement of design and build processes can be used to create solution elements.

The principle of recursive application and how it relates to life cycle models is described in Life Cycle Models.

Activity Mapping

This topic belongs to the Systems Approach Applied to Engineered Systems KA from Part 2 Foundations of Systems Engineering. Other topics about activities, from the same KA, relate to high level technical processes defined in KAs in Part 3, Systems Engineering and Management, in the following way:

- Identifying and Understanding Problems and Opportunities topic relates to the Concept Definition KA.
- Synthesizing Possible Solutions and Analysis and Selection between Alternative Solutions topics relate to the System Definition KA.
- Implementing and Proving a Solution topic relates to the System Realization KA.
- Deploying, Using, and Sustaining Systems to Solve Problems topic relates to the Product and Service Life Management KA.

Part 3 discusses the principles defined in each of the systems approach activities, and how they help shape the technical processes to which they are mapped.

References

Works Cited

Adcock, R.D. 2005. "Tailoring Systems Engineering Lifecycle Processes to meet the challenges of Project and Programme". *INCOSE International Symposium 2005*. Volume 15. Issue 1.

Boehm, B. and A. Jain. 2006. "A value-based theory of Systems Engineering." Presented at the 16th Annual INCOSE Systems Engineering Conference (Orlando FL).

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

INCOSE. 2011. *INCOSE Systems Engineering Handbook*, version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TP-2003-002-03.2.1.

Kruchten, P. 2003. *The Rational Unified Process: An Introduction,* 3rd edition. Boston, MA: Addison Wesley.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Martin J.N. 1997. *Systems Engineering Guidebook*. Boca Raton, FL, USA: CRC Press.

Martin, J, 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems." INCOSE 2004 - 14th Annual International Symposium Proceedings.

Ring, J. 1998. "A Value Seeking Approach to the Engineering of Systems." Proceedings of the IEEE Conference on Systems, Man, and Cybernetics. p. 2704-2708.

Senge, P. 1990. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York, NY, USA: Doubleday/Currency.

Primary References

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College.

Additional References

Blanchard, B. and W.J. Fabrycky 2006. *Systems Engineering and Analysis.* Upper Saddle River, NJ: Prentice Hall.

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

< Previous Article | Parent Article | Next Article (Part 3)>

SEBoK v. 1.9.1, released 16 October 2018

Part 3: Systems Engineering and Management

Systems Engineering and Management

Part 3 of the Guide to the SE Body of Knowledge (SEBoK) focuses on the general knowledge of *how* systems are engineered.

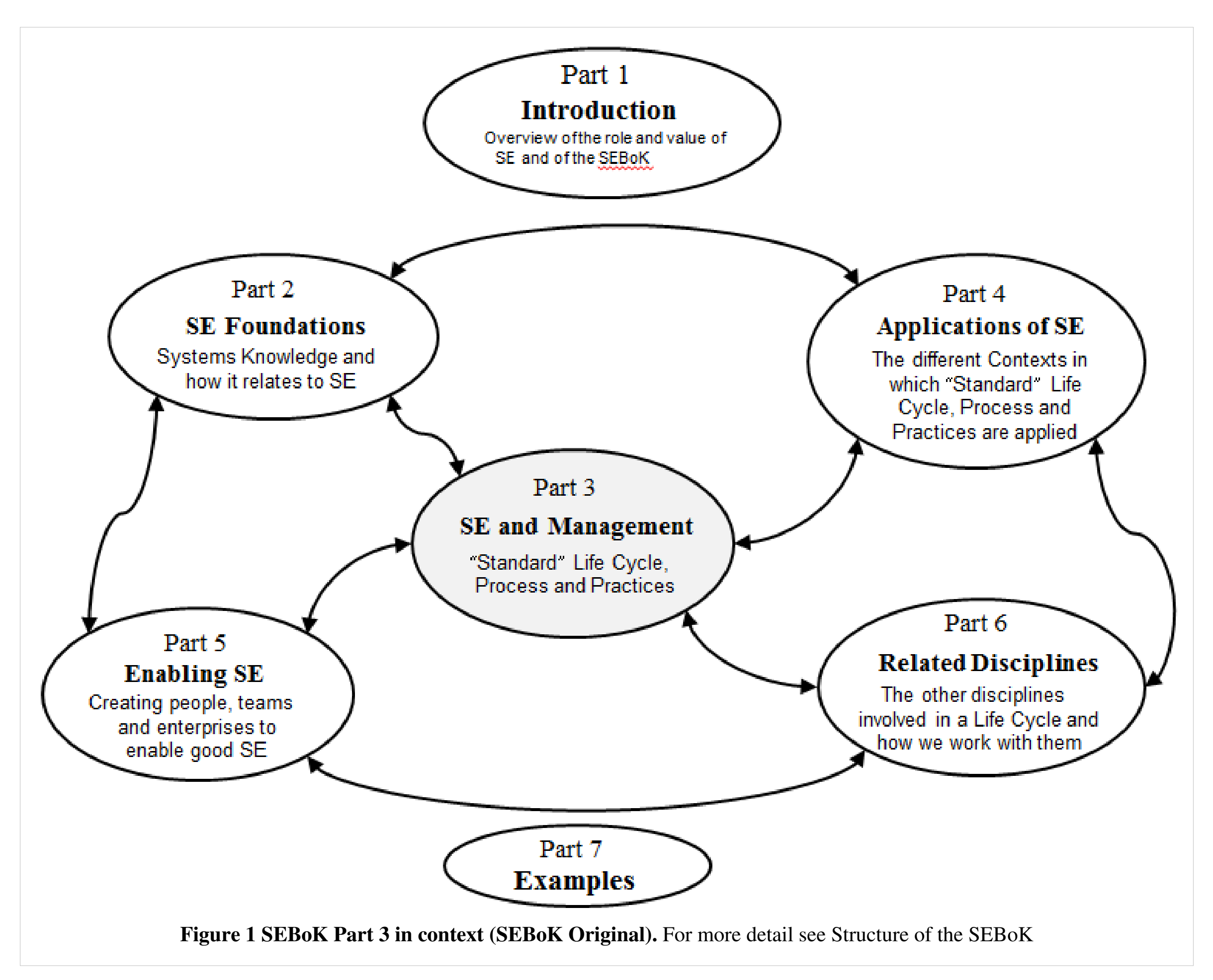


Figure 1 SEBoK Part 3 in context (SEBoK Original). For more detail see Structure of the SEBoK

This part builds upon Part 2: Foundations of Systems Engineering, which discusses the need for a Systems Approach (glossary) applied to one or more Engineered System (glossary) contexts as a part of managed interventions into complex real world problems. Part 3 provides an overview of the common uses of life cycle models to organize the technical and none technical aspects of SE and discusses Systems Engineering Management activities. Part 3 also discusses the most commonly-used SE technical processes; provides additional references to the common methods, tools, and techniques used in these processes

The commonly recognized definition of systems engineering (SE) used across the SEBoK (INCOSE 2015) defines SE as an interdisciplinary approach which applies across the complete life cycle of an identified System-of-Interest. The definition states that systems engineering "**integrates all the disciplines and speciality groups into a team effort forming a structured development process that proceeds from concept to production to operation**". Thus, SE is an engineering discipline concerned with all aspects of an engineered systems life, including how we

organize to do the engineering, what is produced by that engineering and how the resulting systems are used and sustained to meet stakeholder needs.

Part 3 provides only an overview of how systems are engineered in a generic sense. Part 4 provides more specific information as to how the principles discussed in Part 3 are applied differently in consideration of product systems, service systems, enterprise systems, and systems of systems (SoS) contexts. Part 5 explains how people and organizations may approach utilizing these principles as part of a holistic systems approach. Part 6 contains references to other engineering and management disciplines, which work with the SE processes within a systems life cycle, but do not fall under the umbrella of SE.

Systems engineering is transitioning to a model-based approach, model-based systems engineering (MBSE), like many other engineering disciplines. The aim is to enhance productivity and quality, and to cope with the design of increasingly complex systems. Although, models have always been used by systems engineering to create information about engineered systems, that information has been translated and managed through document based artifacts. In a model-based approach, the information about the system is captured in a shared system model, made up of a set of integrated models appropriate to the life cycle stages. This model is managed and controlled throughout the system life cycle as noted in Part 2 under Representing Systems with Models. This provides the ability to maintain more consistent, precise, and traceable information about the system. The system model provides an authoritative source of information that can be communicated across the development team and other stakeholders, can be used to generate views of the system relevant to particular stakeholders, and be used to generate documentation about the system similar to more traditional systems engineering documentation. The model can also be analyzed to assess the integrity of the system specification and design. A model also captures knowledge in a way that can be more readily reused than traditional document based approaches. In a model-based systems engineering approach, the processes referred to in this and other Parts of the SEBoK remain fundamentally the same, but the artifacts produced are model-based. Some examples of MBSE methods are highlighted in A Survey of Model-Based Systems Engineering (MBSE) Methodologies (Estefan 2008). It is anticipated that as the transition to model-based practices occurs, the SEBoK will be updated to reflect the body of current and emerging practice.

Knowledge Areas in Part 3

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 3 contains the following knowledge areas:

- Introduction to Life Cycle Processes
- Life Cycle Models
- Concept Definition
- System Definition
- System Realization
- System Deployment and Use
- Systems Engineering Management
- Product and Service Life Management
- Systems Engineering Standards

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

Value of Ontology Concepts for Systems Engineering

Ontology is the set of entities presupposed by a theory (Collins English Dictionary 2011). Systems engineering, and system development in particular, is based on concepts related to mathematics and proven practices. A SE ontology can be defined considering the following path.

SE provides engineers with an approach based on a set of concepts (i.e., stakeholder, requirement, function, scenario, system element, etc.) and generic processes. Each process is composed of a set of activities and tasks gathered logically around a theme or a purpose. A process describes "what to do" using the applied concepts. The implementation of the activities and tasks is supported by methods and modeling techniques, which are composed themselves of elementary tasks; they describe the "how to do" of SE. The activities and tasks of SE are transformations of generic data using predefined concepts. Those generic data are called entities, classes, or types. Each *entity* is characterized by specific *attributes*, and each attribute may have a different value. All along their execution, the activities and tasks of processes, methods, and modeling techniques exchange instances of generic entities according to logical *relationships*. These relationships allow the engineer to link the entities between themselves (traceability) and to follow a logical sequence of the activities and the global progression (engineering management). Cardinality is associated with every relationship, expressing the minimum and maximum number of entities that are required in order to make the relationship valid. Additional information on this subject may be found in *Engineering Complex Systems with Models and Objects* (Oliver, Kelliher, and Keegan 1997).

The set of SE entities and their relationships form an ontology, which is also referred to as an "engineering meta-model". Such an approach is used and defined in the ISO 10303 standard (ISO 2007). There are many benefits to using an ontology. The ontology allows or forces:

- the use of a standardized vocabulary, with carefully chosen names, which helps to avoid the use of synonyms in the processes, methods, and modeling techniques
- the reconciliation of the vocabulary used in different modeling techniques and methods
- the automatic appearance of the traceability requirements when implemented in databases, SE tools or workbenches, and the quick identification of the impacts of modifications in the engineering data set
- the continual observation of the consistency and completeness of engineering data; etc.

Throughout Part 3, there are discussions of the ontological elements specifically relevant to a given topic.

Mapping of Topics to ISO/IEC 15288, System Life Cycle Processes

Figure 2, below, shows the relative position of the KA's of the SEBoK with respect to the processes outlined in the ISO/IEC/IEEE 15288 (ISO 2015) standard.

As shown, all of the major processes described in ISO/IEC/IEE 15288:2015 are discussed within the SEBoK.

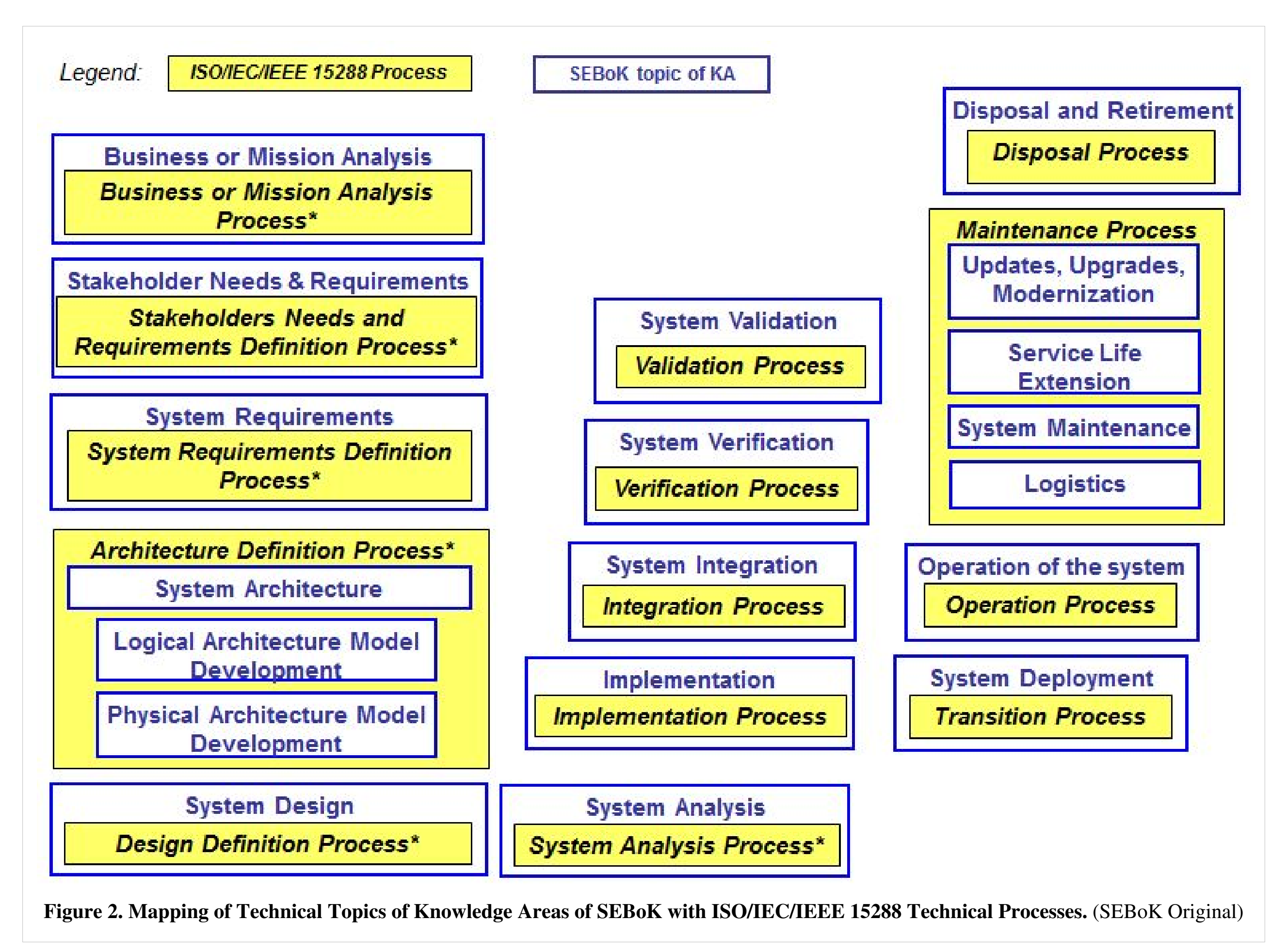


Figure 2. Mapping of Technical Topics of Knowledge Areas of SEBoK with ISO/IEC/IEEE 15288 Technical Processes. (SEBoK Original)

The ISO/IEC/IEEE 15288:2015 marked with an * are new or have been renamed and modified in scope for this revision of the standard.

These changes and associated changes to the SEBoK now mean that the two are significantly more closely aligned than before. It should also be noted that the latest update of the INCOSE SE Handbook (INCOSE 2015) is now fully aligned with the 2015 revision of the standard.

Any future evolution of Life Cycle Process knowledge in the SEBoK will be complementary to these standard descriptions of the generic SE process set.

References

Collins English Dictionary, s.v. "Ontology." 2011.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies*, rev, B. Seattle, WA: International Council on Systems Engineering. INCOSE-TD-2007-003-02. Accessed April 13, 2015 at http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute for Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO. 2007. *Systems Engineering and Design.* Geneva, Switzerland: International Organization for Standardization (ISO). ISO 10303-AP233.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering Complex Systems with Models and Objects*. New York, NY, USA: McGraw-Hill.

Primary References

INCOSE. 2015. *Systems Engineering Handbook* - A Guide for System Life Cycle Processes and Activities", version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. Systems and Software Engineering -- System Life Cycle Processes. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

Additional References

Bell Telephone Laboratories. 1982. *Engineering and Operations in the Bell System*. Murray Hill, NJ: Bell Telephone Laboratories.

Fortescue, P.W., J. Stark, and G. Swinerd. 2003. *Spacecraft Systems Engineering*. New York, NY, USA: J. Wiley.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Introduction to Life Cycle Processes

Introduction to Life Cycle Processes

In this Knowledge Area we introduce key principles of life cycle, life cycle model and life cycle processes. A generic SE paradigm is described; this forms a starting point for discussions of more detailed life cycle knowledge.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Generic Life Cycle Model
- Applying Life Cycle Processes
- Life Cycle Processes and Enterprise Need

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

Life Cycle Terminology

The term Life Cycle (glossary) is one that engineering has borrowed from the natural sciences, it is used to describe both the changes a single organism goes through over it life and how the lives of multiple organisms interact to sustain or evolve a population. We use it in engineering in the same ways to describe the complete life of an instance of a System-of-Interest (glossary) (SoI); and the managed combination of multiple such instances to provide capabilities which deliver stakeholder satisfaction.

A life cycle model identifies the major stages that a specific SoI goes through, from its inception to its retirement. Life cycle models are generally implemented in development projects, and are strongly aligned with management planning and decision making.

Generic Systems Engineering Paradigm

Figure 1 identifies the overall goals of any SE effort, which are: the understanding of stakeholder value; the selection of a specific need to be addressed; the transformation of that need into a system (the product or service that provides for the need); and the use of that product or service to provide the stakeholder value. This paradigm has been developed according to the principles of the systems approach discussed in Part 2 and is used to establish a basis for the KAs in Part 3 and Part 4 of the SEBoK.

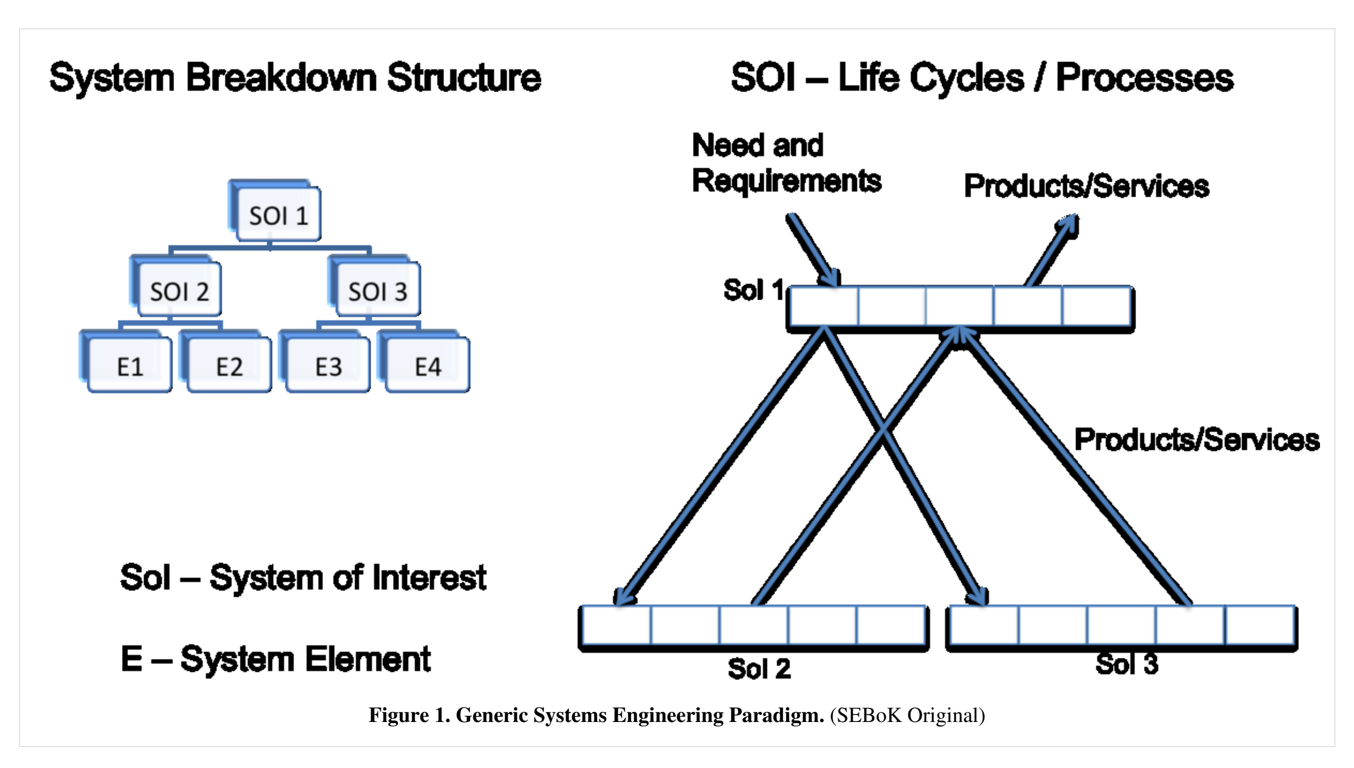


Figure 1. Generic Systems Engineering Paradigm. (SEBoK Original)

On the left side of Figure 1, there are SoI's identified in the formation of a system breakdown structure. SoI 1 is broken down into its basic elements, which in this case are systems as well (SoI 2 and SoI 3). These two systems are composed of system elements that are not refined any further.

On the right side of Figure 1, each SoI has a corresponding life cycle model (glossary) which is composed of stages that are populated with processes. The function of these processes is to define the work that is to be performed and the associated artifacts to be produced. In a model-based approach, these artifacts are captured in the system model that represent the SoI's. Note that some of the requirements defined to meet the need are distributed in the early stages of the life cycle for SoI 1, while others are designated to the life cycles of SoI 2 or SoI 3. The decomposition of the system illustrates the fundamental concept of recursion (glossary) as defined in the ISO/IEC/IEEE 15288 standard; with the standard being reapplied for each SoI (ISO 2015). It is important to point out that the requirements may be allocated to different system elements, which may be integrated in different life cycle stages of any of the three SoI's; however, together they form a cohesive system. For example, SoI 1 may be a simple vehicle composed of a chassis, motor and controls, SoI 2 an embedded hardware system, and Sol 3 a software intensive interface and control system.

When performing SE processes in stages, iteration (glossary) between stages is often required (e.g. in successive refinement of the definition of the system or in providing an update or upgrade of an existing system). The work performed in the processes and stages can be performed in a concurrent manner within the life cycle of any of the systems of interest and also among the multiple life cycles.

This paradigm provides a fundamental framework for understanding generic SE (seen in Part 3), as well as for the application of SE to the various types of systems described in Part 4.

References

Works Cited

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

Primary References

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Generic Life Cycle Model

As discussed in the generic life cycle paradigm in Introduction to Life Cycle Processes each System-of-Interest (glossary) (SoI) has an associated Life Cycle Model (glossary). The generic life cycle model below applies to a single SoI. SE must generally be synchronised across a number tailored instances of such life cycle models to fully satisfy stakeholder needs. More complex life cycle models which address this are described in Life Cycle Models.

A Generic System Life Cycle Model

There is no single "one-size-fits-all" system life cycle model that can provide specific guidance for all project situations. Figure 1, adapted from (Lawson 2010, ISO 2015, and ISO 2010), provides a generic life cycle model that forms a starting point for the most common versions of pre-specified, evolutionary, sequential, opportunistic, and concurrent life cycle processes. The model is defined as a set of stages, within which technical and management activities are performed. The stages are terminated by decision gates where the key stakeholders decide whether to proceed into the next stage, to remain in the current stage, or to terminate or re-scope related projects.

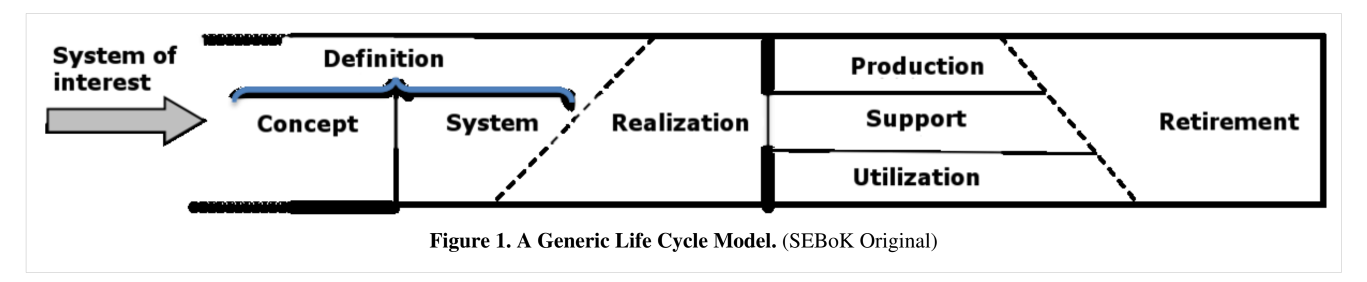


Figure 1. A Generic Life Cycle Model. (SEBoK Original)

Elaborated definitions of these stages are provided below, in the glossary, and in various other ways in subsequent articles.

The **Concept Definition** stage begins with a decision by a protagonist (individual or organization) to invest resources in a new or improved engineered system. Inception begins with a set of stakeholders agreeing to the need for change to an engineered system context and exploring whether new or modified SoI can be developed, in which

the life cycle benefits are worth the investments in the life cycle costs. Activities include: developing the concept of operations and business case; determining the key stakeholders and their desired capabilities; negotiating the stakeholder requirements among the key stakeholders and selecting the system's non-developmental items (NDIs).

The **System Definition** stage begins when the key stakeholders decide that the business needs and stakeholder requirements are sufficiently well defined to justify committing the resources necessary to define a solution options in sufficient detail to answer the life cycle cost question identified in concept definition and provide a basis of system realization if appropriate. Activities include developing system architectures; defining and agreeing levels of system requirements; developing systems-level life cycle plans and performing system analysis in order to illustrate the compatibility and feasibility of the resulting system definition. The transition into the system realization stage can lead to either single-pass or multiple-pass development.

It should be noted that the concept and system definition activities above describe activities performed by *systems engineers* when performing *systems engineering*. There is a very strong concurrency between proposing a problem situation or opportunity and describing one or more possible system solutions, as discussed in Systems Approach Applied to Engineered Systems. Other related definition activities include: prototyping or actual development of high-risk items to show evidence of system feasibility; collaboration with business analysts or performing mission effectiveness analyses to provide a viable business case for proceeding into realization; and modifications to realized systems to improve their production, support or utilization. These activities will generally happen through the system life cycle to handle system evolution, especially under multiple-pass development. This is discussed in more detail in the Life Cycle Models knowledge area.

The **System Realization** stage begins when the key stakeholders decide that the SoI architecture and feasibility evidence are sufficiently low-risk to justify committing the resources necessary to develop and sustain the initial operational capability (IOC) or the single-pass development of the full operational capability (FOC). Activities include: construction of the developmental elements; integration of these with each other and with the non-developmental item (NDI) elements; verification and validation of the elements and their integration as it proceeds; and preparing for the concurrent production, support, and utilization activities.

The **System Production, Support, and Utilization (PSU)** stages begins when the key stakeholders decide that the SoI life-cycle feasibility and safety are at a sufficiently low-risk level that justifies committing the resources necessary to produce, field, support, and utilize the system over its expected lifetime. The lifetimes of production, support, and utilization are likely to be different. After market support will generally continue after production is complete and users will often continue to use unsupported systems.

System Production involves the fabrication of instances or versions of a SoI and of associated after market spare parts. It also includes production quality monitoring and improvement; product or service acceptance activities; and continuous production process improvement. It may include low-rate initial production (LRIP) to mature the production process or to promote the continued preservation of the production capability for future spikes in demand.

Systems Support includes various classes of maintenance: corrective (for defects), adaptive (for interoperability with independently evolving co-dependent systems), and perfective (for enhancement of performance, usability, or other key performance parameters). It also includes hot lines and responders for user or emergency support and the provisioning of needed consumables (gas, water, power, etc.). Its boundaries include some gray areas, such as the boundary between small system enhancements and the development of larger complementary new additions, and the boundary between rework/maintenance of earlier fielded increments in incremental or evolutionary development. *Systems Support* usually continues after *System Production* is terminated.

System Utilization includes the use of the SoI in its context by operators, administrators, the general public, or systems above it in the system-of-interest hierarchy. It usually continues after *Systems Support* is terminated.

The **System Retirement** stage is often executed incrementally as system versions or elements become obsolete or are no longer economical to support and therefore undergo disposal or recycling of their content. Increasingly affordable considerations make system re-purposing an attractive alternative.

Applying the Life Cycle Model

Figure 1 shows just the single-step approach for proceeding through the stages of a SoI life cycle. In Life Cycle Models knowledge area we discuss examples of real world enterprises and their drivers, both technical and organizational. These have lead to a number of documented approaches for sequencing the life cycle stages to deal with some of the issues raised. The Life Cycle Models KA summarizes a number of incremental and evolutionary life cycle models, including their main strengths and weaknesses and also discusses criteria for choosing the best-fit approach.

In figure 1 we have listed key technical and management activities critical to successful completion of each stage. This is a useful way to illustrate the goals of each stage and gives an indication of how processes align with these stages. This can be important when considering how to plan for resources, milestones, etc. However, it is important to observe that the execution of process activities are not compartmentalized to particular life cycle stages (Lawson 2010). In Applying Life Cycle Processes we discuss a number of views on the nature of the inter-relationships between process activities within a life cycle model. In general, the technical and management activities are applied in accordance with the principles of concurrency, Iteration (glossary) and Recursion (glossary) described in the generic life cycle paradigm.

References

Works Cited

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

ISO/IEC. 2010. Systems and Software Engineering, Part 1: Guide for Life Cycle Management. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 24748-1:2010.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Primary References

Forsberg, K., H. Mooz, H. Cotterman. 2005. *Visualizing Project Management*, 3rd Ed. Hoboken, NJ: J. Wiley & Sons.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Applying Life Cycle Processes

The Generic Life Cycle Model describes a set of life cycle stages and their relationships. In defining this we described some of the technical and management activities critical to the success of each stage. While this association of activity to stage is important we must also recognise the through life relationships between these activities to ensure we take a Systems Approach (glossary).

Systems Engineering technical and management activities are defined in a set of life cycle processes. These group together closely related activities and allow us to describe the relationships between them. In this topic we discuss a number of views on the nature of the inter-relationships between process activities within a life cycle model.

In general, the technical and management activities are applied in accordance with the principles of concurrency, Iteration (glossary) and Recursion (glossary) described in the generic systems engineering paradigm. These principles overlap to some extent and can be seen as related views of the same fundamental need to ensure we can take a holistic systems approach, while allowing for some structuring and sequence of our activities. The views presented below should be seen as examples of the ways in which different SE authors present these overlapping ideas.

Life Cycle Process Terminology

Process

A process is a series of actions or steps taken in order to achieve a particular end. Processes can be performed by humans or machines transforming inputs into outputs.

In SEBoK processes are interpreted in several ways, including: technical, life cycle, business, or manufacturing flow processes. Many of the Part 3 sections are structured along technical processes (e.g. design, verification); however, Life Cycle Models also describes a number of high level program life cycle sequence which call themselves processes (e.g. rational unified process (RUP), etc.).

Part 4: Applications of Systems Engineering and Part 5: Enabling Systems Engineering utilize processes that are related to services and business enterprise operations.

Systems Engineering Life Cycle Process (glossary) define technical and management activities performed across one or more stages to provide the information needed to make life cycle decisions; and to enable realization, use and sustainment of a System-of-Interest (glossary) (SoI) across its life cycle model as necessary. This relationship between Life Cycle Model (glossary) and process activities can be used to describe how SE is applied to different system contexts.

Requirement

A requirement is something that is needed or wanted, but may not be compulsory in all circumstances. Requirements may refer to product or process characteristics or constraints. Different understandings of requirements are dependent on process state, level of abstraction, and type (e.g. functional, performance, constraint). An individual requirement may also have multiple interpretations over time.

Requirements exist at multiple levels of enterprise or system with multiple levels abstraction. This ranges from the highest level of the enterprise capability or customer need to the lowest level of the system design .Thus, requirements need to be defined at the appropriate level of detail for the level of the entity to which it applies. See the article Life Cycle Processes and Enterprise Needs for further detail on the transformation of needs and requirements from the enterprise to the lowest system element across concept definition and system definition.

Architecture

An architecture refers to the organizational structure of a system, whereby the system can be defined in different contexts. Architecting is the art or practice of designing the structures. See below for further discussions on the use of levels of Logical and Physical architecture models to define related system and system elements; and support the requirements activities.

Architectures can apply for a system product, enterprise, or service. For example, Part 3 mostly considers product or service related architectures that systems engineers create, but enterprise architecture describes the structure of an organization. Part 5: Enabling Systems Engineering interprets enterprise architecture in a much broader manner than an IT system used across an organization, which is a specific instance of architecture.

Frameworks are closely related to architectures, as they are ways of representing architectures. See the glossary of terms Architecture Framework for definition and examples.

Other Processes

A number of other life cycle processes are mentioned below, including System Analysis (glossary), Integration (glossary), Verification (glossary), Validation (glossary), deployment, operation, Maintenance (glossary) and Disposal (glossary) are discussed in detail in the System Realization and System Deployment and Use knowledge areas.

Life Cycle Process Concurrency

In the Generic Life Cycle Model we have listed key activities critical to successful completion of each stage. This is a useful way to illustrate the goals of each stage and gives an indication of how processes align with these stages. This can be important when considering how to plan for resources, milestones, etc. However, it is important to observe that the execution of process activities are not compartmentalized to particular life cycle stages (Lawson 2010).

Figure 1 shows a simple illustration of the through life nature of technical and management processes. This figure builds directly on the "hump diagram" principles described in Systems Approach Applied to Engineered Systems.

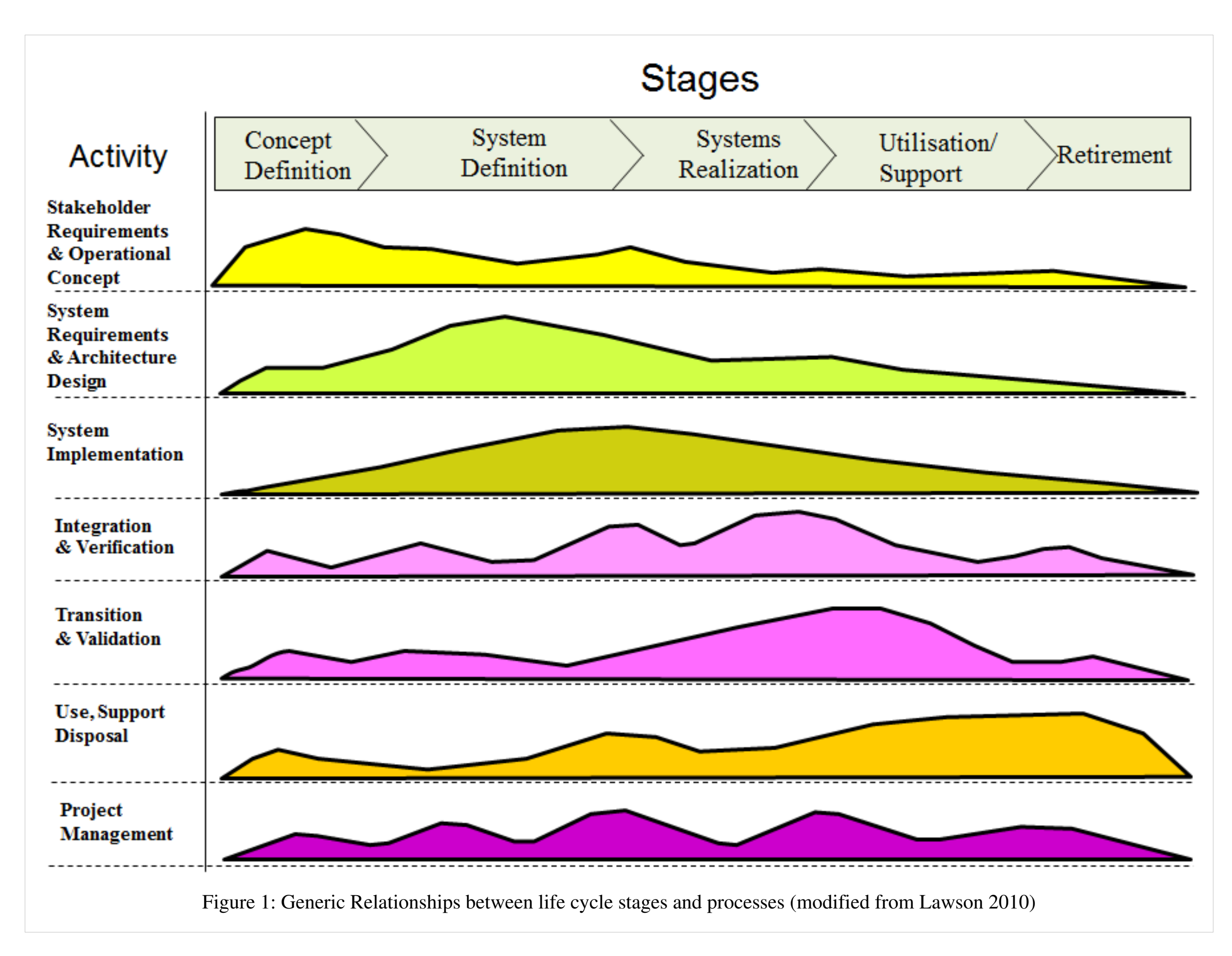


Figure 1: Generic Relationships between life cycle stages and processes (modified from Lawson 2010)

The lines on this diagram represent the amount of activity for each process over the generic life cycle. The peaks (or humps) of activity represent the periods when a process activity becomes the main focus of a stage. The activity before and after these peaks may represent through life issues raised by a process focus, e.g. how will likely maintenance constraints be represented in the system requirements. These considerations help maintain a more holistic perspective in each stage; or they can represent forward planning to ensure the resources needed to complete future activities have been included in estimates and plans, e.g. are all resources need for verification in place or available. Ensuring this hump diagram principle is implemented in a way which is achievable, affordable and appropriate to the situation is a critical driver for all life cycle models.

Life Cycle Process Iteration

The concept of iteration applies to life cycle stages within a life cycle model, and also applies to processes. Figure 2 below gives an example of iteration in the life cycle processes associated with concept and system definition.

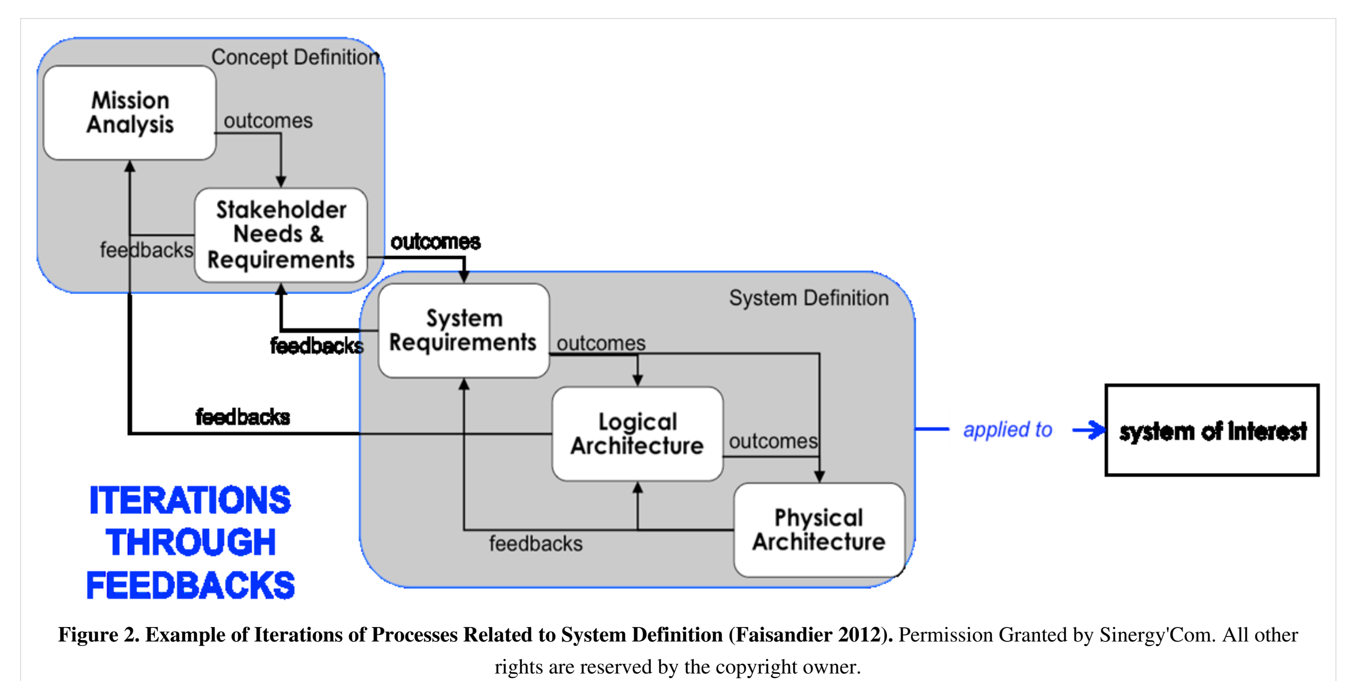


Figure 2. Example of Iterations of Processes Related to System Definition (Faisandier 2012). Permission Granted by Sinergy'Com. All other rights are reserved by the copyright owner.

There is generally a close coupling between the exploration of a problem or opportunity and the definition of one or more feasible solutions, see Systems Approach to Engineered Systems. Thus the related processes in this example will normally be applied in an iterative way. The relationships between these process are further discussed in the System Definition KA.



Figure 3 below gives an example of the iteration between the other life cycle processes.

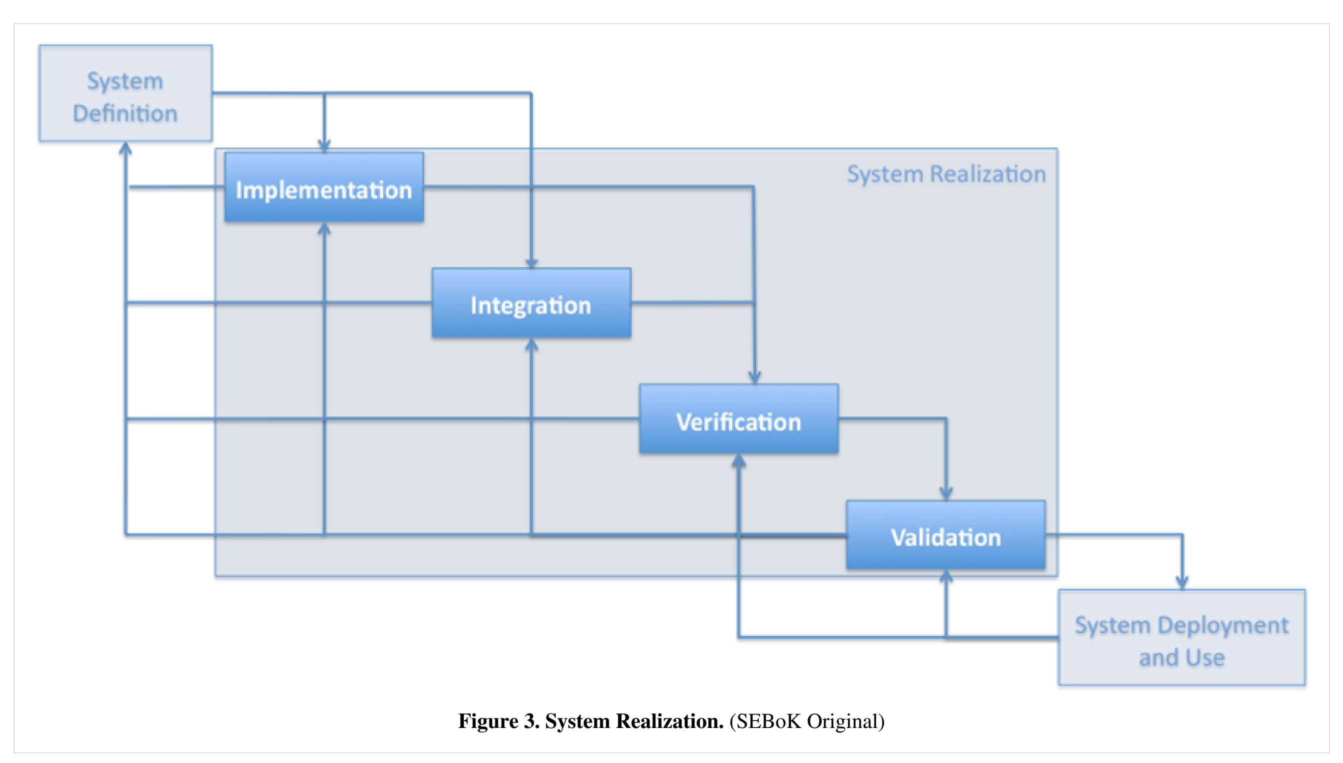


Figure 3. System Realization. (SEBoK Original)

The iterations in this example relate to the overlaps in process outcomes shown in Figure 1. They either allow consideration of cross process issues to influence the system definition, e.g. considering likely integration or verification approaches might make us think about failure modes or add data collection or monitoring elements into the system. Or they allow risk management and through life planning activities to identify the need for future activities.

The relationships between these processes are further discussed in system realization and system deployment and use.

Life Cycle Process Recursion

The comprehensive definition of a SoI is generally achieved using decomposition layers and system elements (glossary). Figure 4 presents a fundamental schema of a system breakdown structure.

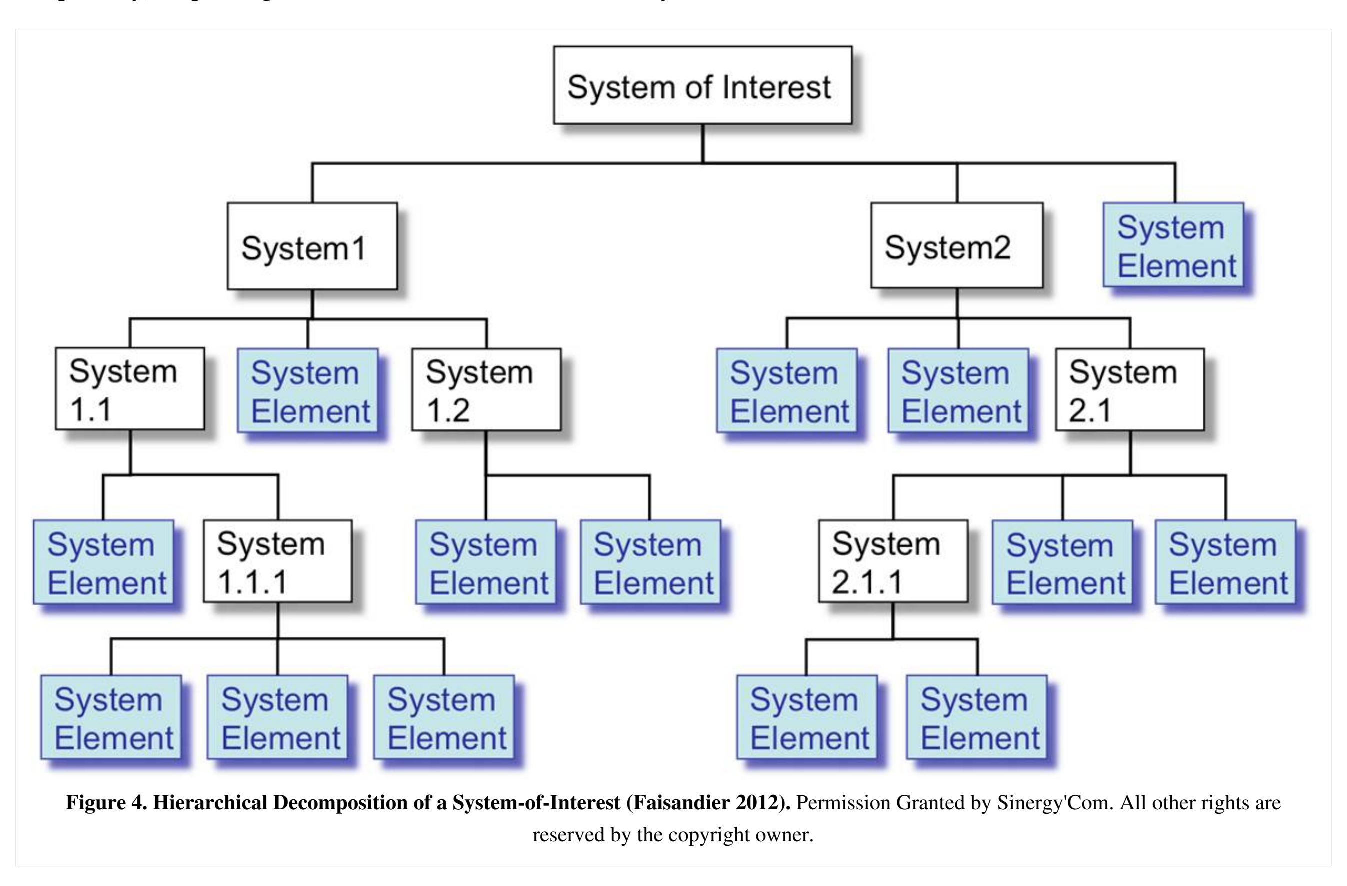


Figure 4. Hierarchical Decomposition of a System-of-Interest (Faisandier 2012). Permission Granted by Sinergy'Com. All other rights are reserved by the copyright owner.

In each decomposition layer and for each system, the System Definition processes are applied recursively because the notion of "system" is in itself recursive; the notions of SoI, system, and system element are based on the same concepts (see Part 2). Figure 5 shows an example of the recursion of life cycle processes.

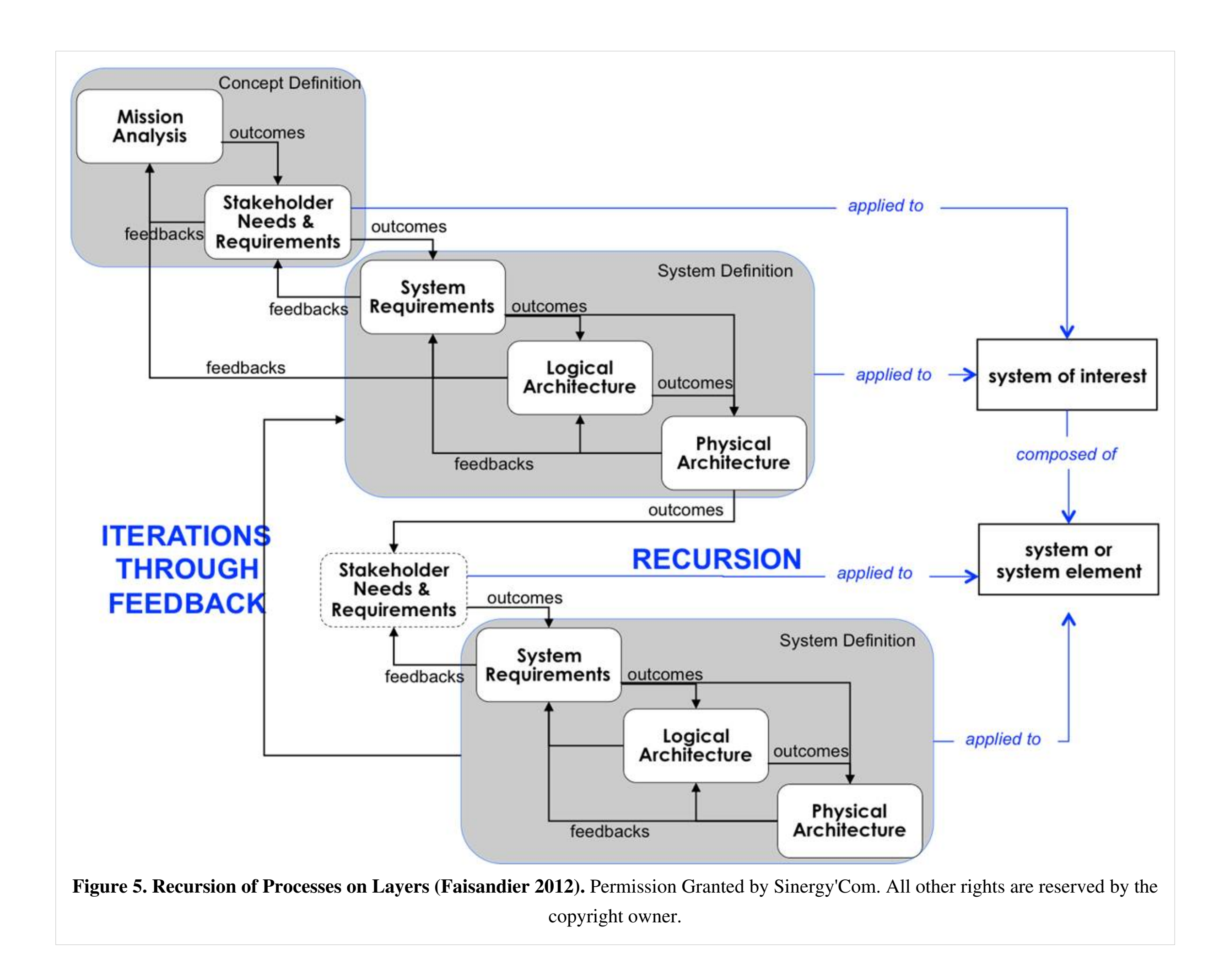


Figure 5. Recursion of Processes on Layers (Faisandier 2012). Permission Granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Systems Approach to Solution Synthesis

The sections above give different perspectives on how SE life cycle processes are related and how this shapes their application. Solution synthesis is described in Part 2 as away of taking a systems approach to creating solution. Synthesis is in general a mixture of top down and bottom up approaches as discussed below.

Top-Down Approach: From Problem to Solution

In a top-down approach, concept definition activities are focused primarily on understanding the problem, the operational needs/requirements within the problem space, and the conditions that constrain the solution and bound the solution space. The concept definition activities determine the mission context, Mission Analysis, and the needs to be fulfilled in that context by a new or modified system (i.e. the SoI), and addresses stakeholder needs and requirements.

The System Definition activities consider functional, behavioral, temporal, and physical aspects of one or more solutions based on the results of concept definition. System Analysis considers the advantages and disadvantages of the proposed system solutions both in terms of how they satisfy the needs established in concept definition, as well as the relative cost, time scales and other development issues. This may require further refinement of the concept definition to ensure all legacy relationships and stakeholders relevant to a particular solution architecture have been considered in the stakeholder requirements.

The outcomes of this iteration between Concept Definition and System Definition define a required system solution and its associated problem context, which are used for System Realization, System Deployment and Use, and Product and Service Life Management of one or more solution implementations. In this approach problem

understanding and solution selection activities are completed in the front-end portion of system development and design and then maintained and refined as necessary throughout the life cycle of any resulting solution systems. Top-down activities can be sequential, iterative, recursive or evolutionary depending upon the life cycle model.

Bottom-Up Approach: Evolution of the Solution

In some situations, the concept definition activities determine the need to evolve existing capabilities or add new capabilities to an existing system. During the concept definition, the alternatives to address the needs are evaluated. Engineers are then led to reconsider the system definition in order to modify or adapt some structural, functional, behavioral, or temporal properties during the product (glossary) or service (glossary) life cycle for a changing context (glossary) of use or for the purpose of improving existing solutions.

Reverse engineering is often necessary to enable system engineers to (re)characterize the properties of the system-of-interest (SoI) or its elements. This is an important step to ensure that system engineers understand the SoI before beginning modification. For more information on system definition, see the System Definition article.

A bottom-up approach is necessary for analysis purposes, or for (re)using existing elements in the design architecture. Changes in the context of use or a need for improvement can prompt this. In contrast, a top-down approach is generally used to define an initial design solution corresponding to a problem or a set of needs.

Solution Synthesis

In most real problems a combination of bottom-up and top-down approaches provides the right mixture of innovative solution thinking driven by need and constrained and pragmatic thinking driven by what already exists. This is often referred to as a “middle-out” approach.

As well as being the most pragmatic approach, synthesis has the potential to keep the life cycle focused on whole system issues, while allow the exploration of the focused levels of detail needed to describe realizable solutions, see Synthesising System Solutions.

References

Works Cited

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Primary References

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Life Cycle Processes and Enterprise Need

The Generic Life Cycle Model describes a simple translation from a need to achieve an outcome to a proposal to realize a new or modified engineered system. This then forms the basis for the decision to invest time, money and other resources in the life cycle of that System-of-Interest (glossary) (SoI). A significant proportion of the SE activities involved in this transformation involve varying levels and types of Requirement (glossary).

Requirements and Life Cycle

During the concept definition stage of a life cycle, as the enterprise identifies new capabilities that are desired, the business or mission analysis develops a high level set of strategies and need (which may be expressed as mission or business requirements) that reflect the problem space perspective. As the solution space is explored and solution classes are characterized, stakeholder needs are developed and transformed into stakeholder requirements (from a user perspective). After the solution class has been determined and the specific solution is sought, during the system definition stage of a life cycle, the stakeholder requirements are transformed into system requirements (from a solution perspective). As the system definition recursively defines the lower level detail of the solution, requirements are defined with lower levels of abstraction. At the highest level, the ideal requirement is implementation-independent, and therefore not specific to a solution, allowing for a range of possible solutions. At the lowest level, requirement statements may become more specific to the selected solution.

Concept Definition has further descriptions of business or enterprise and stakeholder needs and requirements. It discusses how the new capability for the business or enterprise is defined as part of the understanding of the problem space. It also discusses the development of the stakeholder needs and their transformation into requirements from the user perspective.

System Definition has further descriptions of requirements and their types (e.g. system requirement, stakeholder requirement, derived requirement). It discusses how different process roles/ states, levels, and the nature of requirements apply to the understanding of requirements.

Transforming Enterprise Needs to Requirements

Needs and requirements can exist at a number of levels and the terminology used to describe these levels will vary between application domains and the enterprise which serve them. This can make it difficult to associate generic SE life cycle processes with them. Ryan (Ryan, 2013) proposes a generic model (see Figure 1) in which an Enterprise (glossary) of some kind forms a focus for translating strategic intentions into a system definition. There is an enterprise view in which enterprise leadership sets the enterprise strategies, concepts and plans; a business management view in which business management derive business needs and constraints as well as formalize their requirements; a business operations view in which stakeholders define their needs and requirements. In the systems view an Engineered System (glossary) is defined, expanding to views at the lower-level of system elements if needed. Note that an engineered system may comprise a number of elements including products, people, and processes. A system Architecture (glossary) can be created to define logical and physical views of how these elements together enable a needed capability for the organization.

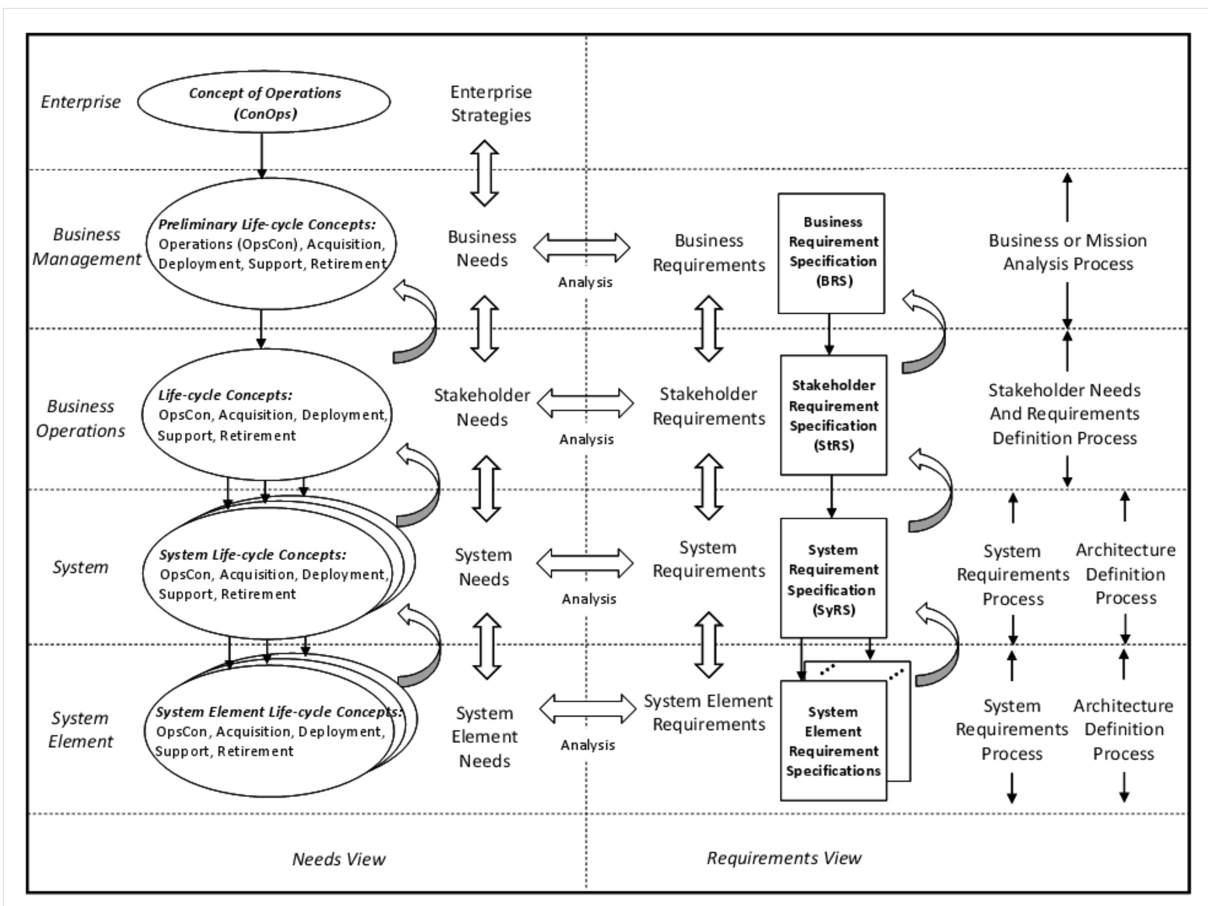


Figure 1. Transformation of needs into requirements (Ryan, 2013) Permission granted by M. Ryan. All other rights are reserved by the copyright owner.

In the following discussion Ryan further defines a set of general terminology to define levels of need and requirements and associates them with generic organizational roles. For a discussion of how this general description might map onto different application contexts or organisational structures see Part 4 and Part 5 respectively.

The various views in Figure 1 are referred to as layers. At the highest layer, the enterprise has a number of strategies that will guide its future. In the illustration above for example, a system has its genesis in a Concept of Operations (ConOps) or Strategic Business Plan (SBP) that communicates the leadership's intentions with regard to the operation of the organization in terms of existing systems and systems to be developed. At this layer the ConOps, or SBP, defines the enterprise in terms of 'brand' and establishes a mission statement and corresponding goals and objectives, which clearly state the reason for the enterprise and its strategy for moving forward.

The Business or Mission Analysis Process begins with the organization's mission or vision statement, goals and objectives communicated by the ConOps or SBP. Business management uses this guidance to define business needs, largely in the form of a life-cycle concepts, which capture the business management's concepts for acquisition, development, marketing, operations, deployment, support, and retirement. These concepts are then used to define specific needs for that layer.

The business needs contained in the life-cycle concepts are elaborated and formalized into business requirements, which are documented in the Business Requirements Specification (BRS) or Business Requirement Document (BRD). The process by which business needs are transformed into business requirements is called mission analysis or business analysis.

Once business management is satisfied their needs and requirements are reasonably complete, they pass them on to the business operations layer. Here, the Stakeholder Needs and Requirements (SNR) Definition Process uses the ConOps or SBP and concepts contained in the life-cycle concepts as guidance. The Requirements Engineer (RE) or

Business Analyst (BA) leads stakeholders from the business operations layer through a structured process to elicit stakeholder needs—in the form of a refined OpsCon or similar document and other life-cycle concepts (see Figure 1). The RE or BA uses a structured process to elicit specific needs, as documented in user stories, use cases, scenarios, system concepts, or operational concepts. For further discussion of the Concept of Operations and the Operational Concept Document, and their interplay, see ANSI/AIAA G-043-2012e, Guide to the Preparation of Operational Concept Documents.

Stakeholder needs are then transformed into a formal set of Stakeholder Requirements, which are documented in the Stakeholder Requirement Specification (StRS) or Stakeholder Requirement Document (StRD). That transformation is guided by a well-defined, repeatable, rigorous, and documented process of requirements analysis. This requirements analysis may involve the use of functional flow diagrams, timeline analysis, N2 Diagrams, design reference missions, modeling and simulations, movies, pictures, states and modes analysis, fault tree analysis, failure modes and effects analysis, and trade studies. In some cases these requirements analysis methods may make use of views created as part of a high level Logical Architecture (glossary).

At the system layer, in the System Requirements Definition Process, the requirements in the StRS are then transformed by the RE or BA into System Requirements, which are documented in the System Requirement Specification (SyRS) or System Requirement Document (SyRD). As in the previous process, the RE or BA accomplishes the transformation of needs into requirements using the same requirements analysis methods described above to define the requirements. At each layer, the resulting requirements will be documented, agreed-to, baselined, and will be put under configuration management. As above the system requirements analysis may also be linked to appropriate logical and Physical Architecture (glossary), either informally or under shared configuration control. Note that some organizations may prepare individual life-cycle concepts for each of a number of systems that are developed to meet the business needs.

Once a set of requirements has been documented, agreed-to, and baselined at one layer they will flow down to the next layer as shown in Figure 1. At the next layer, the requirements are a result of the transformation process of the needs at that layer as well a result of the decomposition or derivation of the requirements from the previous layer. As such, a number of SyRS or SyRD requirements may be either decomposed from (that is, made explicit by the requirements of) or derived from (that is, implied by the requirements of) the StRS or StRD. The same is true at the subsystem or system element layer, where a number of the subsystem or system element requirements may be either decomposed or derived from the SyRS or SyRD. In all cases, for each layer shown in Figure 1, the set of requirements can be traced back to the requirements at the previous layer from which they were either decomposed or derived. This process continues for the next layer of system elements.

How requirements are expressed differs through these layers, and therefore so do the rules for expressing them. As requirements are developed – and solutions designed – down through the layers of abstraction, we expect statements of requirement to become more and more specific. At the highest level, the ideal requirement is not specific to a particular solution, and permits a range of possible solutions. At the lowest level, statements of requirement will be entirely specific to the selected solution. It is important to note that the form of requirements at one layer may not be appropriate for another layer. For example, at the business management layer, there may be a requirement that all products are “safe”. While this is a poor system requirement, it is appropriate for the Business Management layer. At the next layer, business operations, there will be less ambiguous and more detailed requirements that define safe. These requirements apply across all product lines. At the system layer, more specific safety requirements will be developed for that specific system. These requirements will then be allocated to the system elements at the next lower layer.

References

Works Cited

ANSI/AIAA G-043-2012e, Guide to the Preparation of Operational Concept Documents.

Dick, J. and J. Chard, "The Systems Engineering Sandwich: Combining Requirements, Models and Design", INCOSE International Symposium IS2004, July 2004.

Hull, E., K. Jackson, J. Dick, Requirements Engineering, Springer, 2010.

Ryan, M.J., "An Improved Taxonomy for Major Needs and Requirements Artefacts", INCOSE International Symposium IS2013, June 2013.

Primary References

Hull, E., K. Jackson, J. Dick, Requirements Engineering, Springer, 2010.

Ryan, M.J., "An Improved Taxonomy for Major Needs and Requirements Artefacts", INCOSE International Symposium IS2013, June 2013.

Additional References

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Life Cycle Models

Life Cycle Models

The life cycle model is one of the key concepts of systems engineering (SE). A life cycle (glossary) for a system (glossary) generally consists of a series of stages (glossary) regulated by a set of management decisions which confirm that the system is mature enough to leave one stage and enter another.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Life Cycle Process Drivers and Choices
- System Life Cycle Process Models: Vee
- System Life Cycle Process Models: Iterative
- Integration of Process and Product Models
- Lean Engineering

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

Type of Value Added Products/Services

The Generic Life Cycle Model shows just the single-step approach for proceeding through the stages of a system's life cycle. Adding value (as a product, a service, or both), is a shared purpose among all enterprises, whether public or private, for profit or non-profit. Value is produced by providing and integrating the elements of a system into a product or service according to the system description and transitioning it into productive use. These value considerations will lead to various forms of the generic life cycle management approach in Figure 1. Some examples are as follows (Lawson 2010):

- A manufacturing enterprise produces nuts, bolts, and lock washer products and then sells their products as value added elements to be used by other enterprises; in turn, these enterprises integrate these products into their more encompassing value added system, such as an aircraft or an automobile. Their requirements will generally be pre-specified by the customer or by industry standards.
- A wholesaling or retailing enterprise offers products to their customers. Its customers (individuals or enterprises) acquire the products and use them as elements in their systems. The enterprise support system will likely evolve opportunistically, as new infrastructure capabilities or demand patterns emerge.
- A commercial service enterprise such as a bank sells a variety of *products* as services to their customers; for example, this includes current accounts, savings accounts, loans, and investment management. These services add value and are incorporated into customer systems of individuals or enterprises. The service enterprise's support system will also likely evolve opportunistically, as new infrastructure capabilities or demand patterns emerge.
- A governmental service enterprise provides citizens with services that vary widely, but may include services such as health care, highways and roads, pensions, law enforcement, or defense. Where appropriate, these services become infrastructure elements utilized in larger encompassing systems of interest to individuals and/or enterprises. Major initiatives, such as a next-generation air traffic control system or a metropolitan-area crisis management system (hurricane, typhoon, earthquake, tsunami, flood, fire), will be sufficiently complex enough to

follow an evolutionary development and fielding approach. At the element level, there will likely be pre-specified single-pass life cycles.

- For aircraft and automotive systems, there would likely be a pre-specified multiple-pass life cycle to capitalize on early capabilities in the first pass, but architected to add further value-adding capabilities in later passes.
- A diversified software development enterprise provides software products that meet stakeholder requirements (needs), thus providing services to product users. It will need to be developed to have capabilities that can be tailored to be utilized in different customers' life-cycle approaches and also with product-line capabilities that can be quickly and easily applied to similar customer system developments. Its business model may also include providing the customer with system life-cycle support and evolution capabilities.

Within these examples, there are systems that remain stable over reasonably long periods of time and those that change rapidly. The diversity represented by these examples and their processes illustrate why there is no one-size-fits-all process that can be used to define a specific systems life cycle. Management and leadership approaches must consider the type of systems involved, their longevity, and the need for rapid adaptation to unforeseen changes, whether in competition, technology, leadership, or mission priorities. In turn, the management and leadership approaches impact the type and number of life cycle models that are deployed as well as the processes that will be used within any particular life cycle.

There are several incremental and evolutionary approaches for sequencing the life cycle stages to deal with some of the issues raised above. The Life Cycle Models knowledge area summarizes a number of incremental and evolutionary life cycle models, including their main strengths and weaknesses and also discusses criteria for choosing the best-fit approach.

Categories of Life Cycle Model

The Generic System Life Cycle Model in Figure 1 does not explicitly fit all situations. A simple, precedential, follow-on system may need only one phase in the definition stage, while a complex system may need more than two. With build-upon (vs. throwaway) prototypes, a good deal of development may occur during the definition stage. System integration, verification, and validation may follow implementation or acquisition of the system elements. With software test-first and daily builds in particular, integration, verification, and validation are interwoven with element implementation. Additionally, with the upcoming *Third Industrial Revolution* of three-dimensional printing and digital manufacturing (Whadcock 2012), not only initial development but also initial production may be done during the concept stage.

Software is a flexible and malleable medium which facilitates iterative analysis, design, construction, verification, and validation to a greater degree than is usually possible for the purely physical components of a system. Each repetition of an iterative development model adds material (code) to the growing software base, in which the expanded code base is tested, reworked as necessary, and demonstrated to satisfy the requirements for the baseline.

Software can be electronically bought, sold, delivered, and upgraded anywhere in the world within reach of digital communication, making its logistics significantly different and more cost-effective than hardware. It doesn't wear out and its fixes change its content and behavior, making regression testing more complex than with hardware fixes. Its discrete nature provides that its testing cannot count on analytic continuity as with hardware. Adding 1 to 32767 in a 15-bit register does not produce 32768, but 0 instead, as experienced in serious situations, such as with the use of the Patriot Missile.

There are a large number of potential life cycle process models. They fall into three major categories:

1. primarily pre-specified and sequential processes (e.g. the single-step waterfall model)
2. primarily evolutionary and concurrent processes (e.g. lean development, the rational unified process, and various forms of the vee and spiral models)

3. primarily interpersonal and emergent processes (e.g. agile development, scrum, extreme programming (XP), the dynamic system development method, and innovation-based processes)

The emergence of integrated, interactive hardware-software systems made pre-specified processes potentially harmful, as the most effective human-system interfaces tended to emerge with its use, leading to further process variations, such as soft SE (Warfield 1976, Checkland 1981) and human-system integration processes (Booher 2003, Pew and Mavor 2007). Until recently, process standards and maturity models have tried to cover every eventuality and have included extensive processes for acquisition management, source selection, reviews and audits, quality assurance, configuration management, and document management, which in many instances would become overly bureaucratic and inefficient. This led to the introduction of more lean (Ohno 1988; Womack et al. 1990; Oppenheim 2011) and agile (Beck 1999; Anderson 2010) approaches to concurrent hardware-software-human factors approaches such as the concurrent vee models (Forsberg 1991; Forsberg 2005) and Incremental Commitment Spiral Model (Pew and Mavor 2007; Boehm and Lane 2007).

In the next article on System Life Cycle Process Drivers and Choices, these variations on the theme of life cycle models will be identified and presented.

Systems Engineering Responsibility

Regardless of the life cycle models deployed, the role of the systems engineer encompasses the entire life cycle of the system-of-interest. Systems engineers orchestrate the development and evolution of a solution, from defining requirements through operation and ultimately until system retirement. They assure that domain experts are properly involved, all advantageous opportunities are pursued, and all significant risks are identified and, when possible, mitigated. The systems engineer works closely with the project manager in tailoring the generic life cycle, including key decision gates, to meet the needs of their specific project.

Systems engineering tasks are usually concentrated at the beginning of the life cycle; however, both commercial and government organizations recognize the need for SE throughout the system's life cycle. Often this ongoing effort is to modify or change a system, product or service after it enters production or is placed in operation. Consequently, SE is an important part of all life cycle stages. During the production, support, and utilization (PSU) stages, for example, SE executes performance analysis, interface monitoring, failure analysis, logistics analysis, tracking, and analysis of proposed changes. All of these activities are essential to ongoing support of the system.

All project managers must ensure that the business aspect (cost, schedule, and value) and the technical aspect of the project cycle remain synchronized. Often, the technical aspect drives the project and it is the systems engineers' responsibility to ensure that the technical solutions that are being considered are consistent with the cost and schedule objectives. This can require working with the users and customers to revise objectives to fit within the business bounds. These issues also drive the need for decision gates to be appropriately spaced throughout the project cycle. Although the nature of these decision gates will vary by the major categories above, each will involve in-process validation between the developers and the end users. In-process validation asks the question: *"Will what we are planning or creating satisfy the stakeholders' needs?"* In-process validation begins at the initialization of the project during user needs discovery and continues through daily activities, formal decision gate reviews, final product or solution delivery, operations, and ultimately to system closeout and disposal.

References

Works Cited

Anderson, D. 2010. *Kanban.* Sequim, WA: Blue Hole Press.

Beck, K. 1999. *Extreme Programming Explained.* Boston, MA: Addison Wesley.

Boehm, B. and J. Lane. 2007. “Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering.” *CrossTalk*. October 2007: 4-9.

Booher, H. (ed.) 2003. *Handbook of Human Systems Integration*. Hoboken, NJ, USA: Wiley.

Checkland, P. 1999. *Systems Thinking, Systems Practice,* 2nd ed. Hoboken, NJ, USA: Wiley.

Cusumano, M., and D. Yoffie 1998. *Competing on Internet Time*, New York, NY, USA: The Free Press.

Forsberg, K. and H. Mooz. 1991. "The Relationship of System Engineering to the Project Cycle," *Proceedings of NCOSE*, October 1991.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management*, 3rd ed. Hoboken, NJ: J. Wiley & Sons.

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Ohno, T. 1988. *Toyota Production System*. New York, NY: Productivity Press.

Oppenheim, B. 2011. *Lean for Systems Engineering.* Hoboken, NJ: Wiley.

Pew, R. and A. Mavor (eds.). 2007. *Human-System Integration in The System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Warfield, J. 1976. *Systems Engineering*. Washington, DC, USA: US Department of Commerce (DoC).

Whadcock, I. 2012. “A third industrial revolution.” *The Economist.* April 21, 2012.

Womack, J.P., D.T. Jones, and D. Roos 1990. *The Machine That Changed the World: The Story of Lean Production.* New York, NY, USA: Rawson Associates.

Primary References

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Forsberg, K., H. Mooz, H. Cotterman. 2005. *Visualizing Project Management*, 3rd Ed. Hoboken, NJ: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Pew, R. and A. Mavor (eds.). 2007. *Human-System Integration in The System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Additional References

Chrissis, M., M. Konrad, and S. Shrum. 2003. *CMMI: Guidelines for Process Integration and Product Improvement.* New York, NY, USA: Addison Wesley.

Larman , C. and B. Vodde. 2009. *Scaling Lean and Agile Development.* New York, NY, USA: Addison Wesley.

The following three books are not referenced in the SEBoK text, nor are they systems engineering "texts"; however, they contain important systems engineering lessons, and readers of this SEBOK are encouraged to read them.

Kinder, G. 1998. *Ship of Gold in the Deep Blue Sea.* New York, NY, USA: Grove Press.

This is an excellent book that follows an idea from inception to its ultimately successful conclusion. Although systems engineering is not discussed, it is clearly illustrated in the whole process from early project definition to alternate concept development to phased exploration and “thought experiments” to addressing challenges along the way. It also shows the problem of not anticipating critical problems outside the usual project and engineering scope. It took about five years to locate and recover the 24 tons of gold bars and coins from the sunken ship in the 2,500-meter-deep sea, but it took ten years to win the legal battle with the lawyers representing insurance companies who claimed ownership based on 130-year-old policies they issued to the gold owners in 1857.

McCullough, D. 1977. *The Path Between the Seas: The Creation of the Panama Canal (1870 – 1914).* New York, NY, USA: Simon & Schuster.

Although “systems engineering” is not mentioned, this book highlights many systems engineering issues and illustrates the need for SE as a discipline. The book also illustrates the danger of applying a previously successful concept (the sea level canal used in Suez a decade earlier) in a similar but different situation. Ferdinand de Lesseps led both the Suez and Panama projects. It illustrates the danger of not having a fact-based project cycle and meaningful decision gates throughout the project cycle. It also highlights the danger of providing project status without visibility, since after five years into the ten-year project investors were told the project was more than 50 percent complete when in fact only 10 percent of the work was complete. The second round of development under Stevens in 1904 focused on “moving dirt” rather than digging a canal, a systems engineering concept key to the completion of the canal. The Path Between the Seas won the National Book Award for history (1978), the Francis Parkman Prize (1978), the Samuel Eliot Morison Award (1978), and the Cornelius Ryan Award (1977).

Shackleton, Sir E.H. 2008. (Originally published in by William Heinemann, London, 1919). *South: The Last Antarctic Expedition of Shackleton and the Endurance.* Guilford, CT, USA: Lyons Press.

This is the amazing story of the last Antarctic expedition of Shackleton and the *Endurance* in 1914 to 1917. The systems engineering lesson is the continuous, daily risk assessment by the captain, expedition leader, and crew as they lay trapped in the arctic ice for 18 months. All 28 crew members survived.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Life Cycle Process Drivers and Choices

As discussed in the Generic Life Cycle Model article, there are many organizational factors that can impact which life cycle processes are appropriate for a specific system. In addition, technical factors will also influence the types of life cycle models appropriate for a given system. For example, system requirements can be predetermined or they can be changing, depending on the scope and nature of the development for a system. These considerations lead to different life cycle model selections. This article discusses different technical factors which can be considered when selecting a life cycle process model and provides examples, guidance and tools from the literature to support life cycle model selection. The life cycle model selected can impact all other aspects of system design and development. (See the knowledge areas in Part 3 for a description of how the life cycle can impact systems engineering (SE) processes.)

Fixed-Requirements and Evolutionary Development Processes

Aside from the traditional, pre-specified, sequential, single-step development process, there are several models of incremental and evolutionary development; however, there is no one-size-fits-all approach that is best for all situations. For rapid-fielding situations, an easiest-first, prototyping approach may be most appropriate. For enduring systems, an easiest-first approach may produce an unscalable system, in which the architecture is incapable of achieving high levels of performance, safety, or security. In general, system evolution now requires much higher sustained levels of SE effort, earlier and continuous integration and testing, proactive approaches to address sources of system change, greater levels of concurrent engineering, and achievement reviews based on evidence of feasibility versus plans and system descriptions.

Incremental and evolutionary development methods have been in use since the 1960s (and perhaps earlier). They allow a project to provide an initial capability followed by successive deliveries to reach the desired system-of-interest (SoI). This practice is particularly valuable in cases in which

- rapid exploration and implementation of part of the system is desired;
- requirements are unclear from the beginning, or are rapidly changing;
- funding is constrained;
- the customer wishes to hold the SoI open to the possibility of inserting new technology when it becomes mature; and
- experimentation is required to develop successive versions.

In iterative development, each cycle of the iteration subsumes the system elements of the previous iteration and adds new capabilities to the evolving product to create an expanded version of the software. Iterative development processes can provide a number of advantages, including

- continuous integration, verification, and validation of the evolving product;
- frequent demonstrations of progress;
- early detection of defects;
- early warning of process problems; and
- systematic incorporation of the inevitable rework that occurs in software development.

Primary Models of Incremental and Evolutionary Development

The primary models of incremental and evolutionary development focus on different competitive and technical challenges. The time phasing of each model is shown in Figure 1 below in terms of the increment (1, 2, 3, …) content with respect to the definition (Df), development (Dv), and production, support, and utilization (PSU) stages in Figure 1 (A Generic System Life Cycle Model) from the Life Cycle Models article.

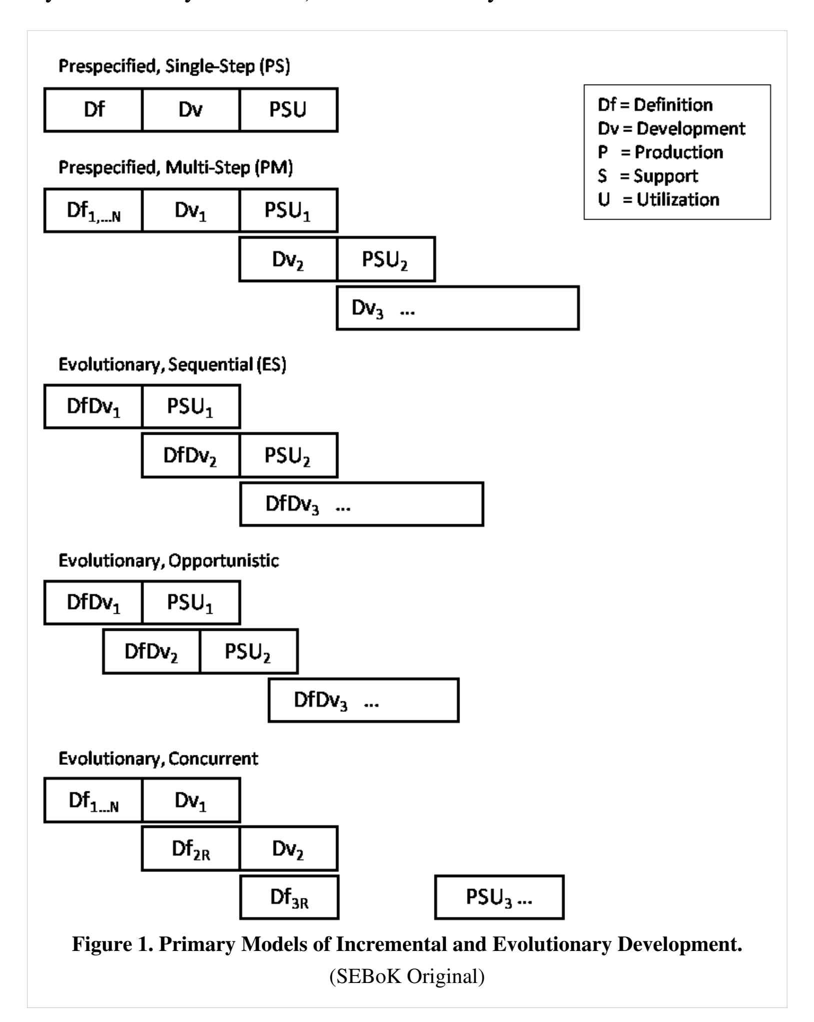


Figure 1. Primary Models of Incremental and Evolutionary Development.
(SEBoK Original)

The Figure 1 notations (Df1..N and Dv1..N) indicate that their initial stages produce specifications not just for the first increment, but for the full set of increments. These are assumed to remain stable for the pre-specified sequential model but are expected to involve changes for the evolutionary concurrent model. The latter's notation (Dv1 and Df2R) in the same time frame, PSU1, Dv2 and Df3R in the same time frame, etc.) indicates that the plans and specifications for the next increment are being re-baselined by a systems engineering team concurrently with the development of the current increment and the PSU of the previous increment. This offloads the work of handling the change traffic from the development team and significantly improves its chances of finishing the current increment on budget and schedule.

In order to select an appropriate life cycle model, it is important to first gain an understanding of the main archetypes and where they are best used. Table 1 summarizes each of the primary models of single-step, incremental and evolutionary development in terms of examples, strengths, and weaknesses, followed by explanatory notes.

Table 1. Primary Models of Incremental and Evolutionary Development (SEBoK Original).

Model	Examples	Pros	Cons
Pre-specified Single-step	Simple manufactured products: Nuts, bolts, simple sensors	Efficient, easy to verify	Difficulties with rapid change, emerging requirements (complex sensors, human-intensive systems)
Pre-specified Multi-step	Vehicle platform plus value-adding pre-planned product improvements (PPPIs)	Early initial capability, scalability when stable	Emergent requirements or rapid change, architecture breakers
Evolutionary Sequential	Small: Agile Larger: Rapid fielding	Adaptability to change, smaller human-intensive systems	Easiest-first, late, costly fixes, systems engineering time gaps, slow for large systems
Evolutionary Opportunistic	Stable development, Maturing technology	Mature technology upgrades	Emergent requirements or rapid change, SysE time gaps
Evolutionary Concurrent	Rapid, emergent development, systems of systems	Emergent requirements or rapid change, stable development increments, SysE continuity	Overkill on small or highly stable systems

The *Pre-specified Single-step* and *Pre-specified Multi-step* models from Table 1 are not evolutionary. Pre-specified multi-step models split the development in order to field an early initial operational capability, followed by several pre-planned product improvements (P3Is). An alternate version splits up the work but does not field the intermediate increments. When requirements are well understood and stable, the pre-specified models enable a strong, predictable process. When requirements are emergent and/or rapidly changing, they often require expensive rework if they lead to undoing architectural commitments.

The*Evolutionary Sequential* model involves an approach in which the initial operational capability for the system is rapidly developed and is upgraded based on operational experience. Pure agile software development fits this model. If something does not turn out as expected and needs to be changed, it will be fixed in thirty days at the time of its next release. Rapid fielding also fits this model for larger or hardware-software systems. Its major strength is to enable quick-response capabilities in the field. For pure agile, the model can fall prey to an easiest-first set of architectural commitments which break when, for example, system developers try to scale up the workload by a factor of ten or to add security as a new feature in a later increment. For rapid fielding, using this model may prove expensive when the quick mash-ups require extensive rework to fix incompatibilities or to accommodate off-nominal usage scenarios, but the rapid results may be worth it.

The *Evolutionary Opportunistic* model can be adopted in cases that involve deferring the next increment until: a sufficiently attractive opportunity presents itself, the desired new technology is mature enough to be added, or until other enablers such as scarce components or key personnel become available. It is also appropriate for synchronizing upgrades of multiple commercial-off-the-shelf (COTS) products. It may be expensive to keep the SE and development teams together while waiting for the enablers, but again, it may be worth it.

The *Evolutionary Concurrent* model involves a team of systems engineers concurrently handling the change traffic and re-baselining the plans and specifications for the next increment, in order to keep the current increment development stabilized. An example and discussion are provided in Table 2, below.

Incremental and Evolutionary Development Decision Table

The Table 2 provides some criteria for deciding which of the processes associated with the primary classes of incremental and evolutionary development models to use.

Table 2. Incremental and Evolutionary Development Decision Table. (Boehm and Lane 2010). Reprinted with permission of the Systems Engineering Research Center. All other rights are reserved by the copyright owner.

Model	Stable, pre-specifiable requirements?	OK to wait for full system to be developed?	Need to wait for next-increment priorities?	Need to wait for next-increment enablers*?
Pre-specified Single-step	Yes	Yes		
Pre-specified Multi-step	Yes	No		
Evolutionary Sequential	No	No	Yes	
Evolutionary Opportunistic	No	No	No	Yes
Evolutionary Concurrent	No	No	No	No

Example enablers: Technology maturity; External-system capabilities; Needed resources; New opportunities

The *Pre-specified Single-step* process exemplified by the traditional waterfall or sequential Vee model is appropriate if the product's requirements are pre-specifiable and have a low probability of significant change and if there is no value or chance to deliver a partial product capability. A good example of this would be the hardware for an earth resources monitoring satellite that would be infeasible to modify after it goes into orbit.

The *Pre-specified Multi-step* process splits up the development in order to field an early initial operational capability and several P3I's. It is best if the product's full capabilities can be specified in advance and are at a low probability of significant change. This is useful in cases when waiting for the full system to be developed incurs a loss of important and deliverable incremental mission capabilities. A good example of this would be a well-understood and well-prioritized sequence of software upgrades for the on-board earth resources monitoring satellite.

The *Evolutionary Sequential* process develops an initial operational capability and upgrades it based on operational experience, as exemplified by agile methods. It is most need in cases when there is a need to get operational feedback on an initial capability before defining and developing the next increment's content. A good example of this would be the software upgrades suggested by experiences with the satellite's payload, such as what kind of multi-spectral data collection and analysis capabilities are best for what kind of agriculture under what weather conditions.

The *Evolutionary Opportunistic* process defers the next increment until its new capabilities are available and mature enough to be added. It is best used when the increment does not need to wait for operational feedback, but it may need to wait for next-increment enablers such as technology maturity, external system capabilities, needed resources, or new value-adding opportunities. A good example of this would be the need to wait for agent-based satellite anomaly trend analysis and mission-adaptation software to become predictably stable before incorporating it into a scheduled increment.

The *Evolutionary Concurrent* process, as realized in the incremental commitment spiral model (Pew and Mavor 2007; Boehm and Lane 2007) and shown in Figure 2, has a continuing team of systems engineers handling the change traffic and re-baselining the plans and specifications for the next increment, while also keeping a development team stabilized for on-time, high-assurance delivery of the current increment and employing a

concurrent verification and validation (V&V) team to perform continuous defect detection to enable even higher assurance levels. A good example of this would be the satellite's ground-based mission control and data handling software's next-increment re-baselining to adapt to new COTS releases and continuing user requests for data processing upgrades.

The satellite example illustrates the various ways in which the complex systems of the future, different parts of the system, and its software may evolve in a number of ways, once again affirming that there is no one-size-fits-all process for software evolution. However, Table 2 can be quite helpful in determining which processes are the best fits for evolving each part of the system and the three-team model in Figure 2 provides a way for projects to develop the challenging software-intensive systems of the future that will need both adaptability to rapid change and high levels of assurance.

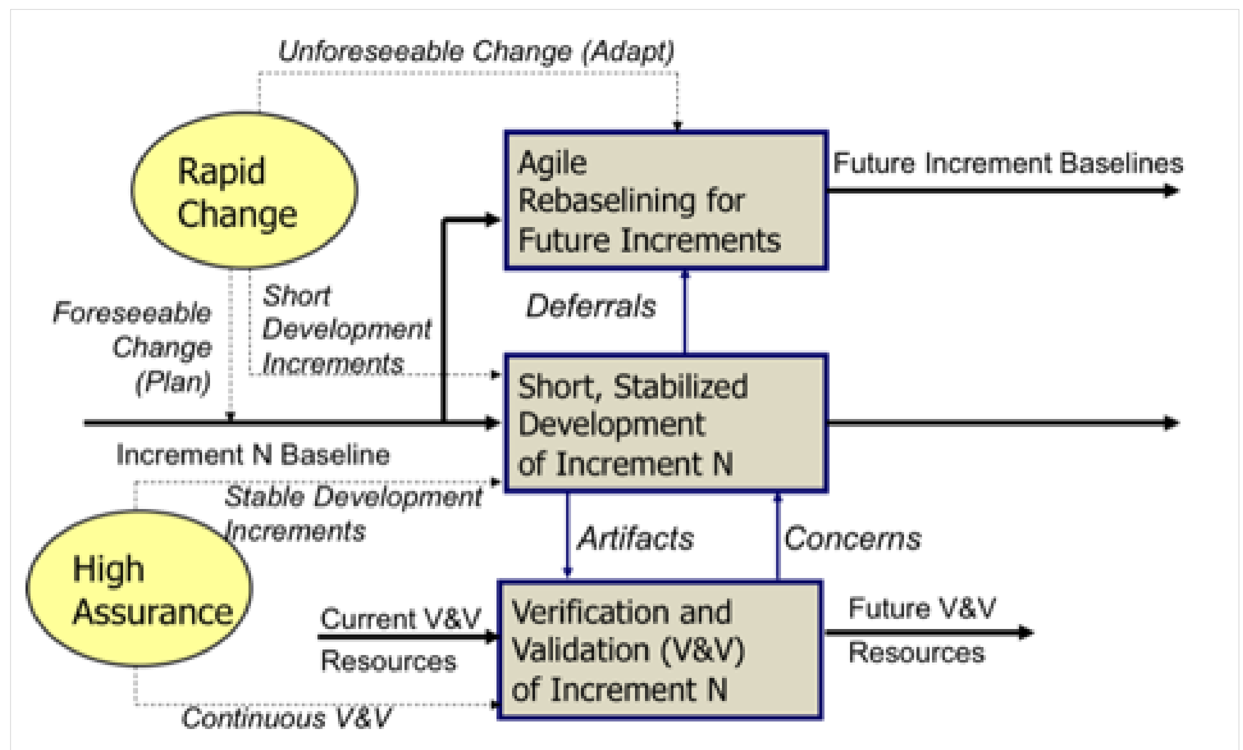


Figure 2. Evolutionary-Concurrent Rapid Change Handling and High Assurance (Pew and Mavor 2007, Figure 2-6). Reprinted with permission from the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

References

Works Cited

Boehm, B. 2006. "Some Future Trends and Implications for Systems and Software Engineering Processes." *Systems Engineering.* 9(1): 1-19.

Boehm, B. and J. Lane. 2007. "Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering." *CrossTalk.* October 2007: 4-9.

Boehm, B. and J. Lane. 2010. *DoD Systems Engineering and Management Implications for Evolutionary Acquisition of Major Defense Systems.* SERC RT-5 report, March 2010. USC-CSSE-2010-500.

Cusumano, M. and D. Yoffee. 1998. *Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft.* New York, NY, USA: Free Press.

Pew, R. and A. Mavor (eds.). 2007. *Human-System Integration in the System Development Process: A New Look.* Washington DC, USA: The National Academies Press.

Primary References

Pew, R., and A. Mavor (eds.). 2007. *Human-System Integration in the System Development Process: A New Look*. Washington, DC, USA: The National Academies Press.

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Life Cycle Process Models: Vee

There are a large number of life cycle process models. As discussed in the System Life Cycle Process Drivers and Choices article, these models fall into three major categories: (1) primarily pre-specified and sequential processes; (2) primarily evolutionary and concurrent processes (e.g., the rational unified process and various forms of the Vee and spiral models); and (3) primarily interpersonal and unconstrained processes (e.g., agile development, Scrum, extreme programming (XP), the dynamic system development method, and innovation-based processes).

This article specifically focuses on the Vee Model as the primary example of pre-specified and sequential processes. In this discussion, it is important to note that the Vee model, and variations of the Vee model, all address the same basic set of systems engineering (SE) activities. The key difference between these models is the way in which they group and represent the aforementioned SE activities.

General implications of using the Vee model for system design and development are discussed below; for a more specific understanding of how this life cycle model impacts systems engineering activities, please see the other knowledge areas (KAs) in Part 3.

A Primarily Pre-specified and Sequential Process Model: The Vee Model

The sequential version of the Vee Model is shown in Figure 1. Its core involves a sequential progression of plans, specifications, and products that are baselined and put under configuration management. The vertical, two-headed arrow enables projects to perform concurrent opportunity and risk analyses, as well as continuous in-process validation. The Vee Model encompasses the first three life cycle stages listed in the "Generic Life Cycle Stages" table of the INCOSE *Systems Engineering Handbook*: exploratory research, concept, and development (INCOSE 2012).

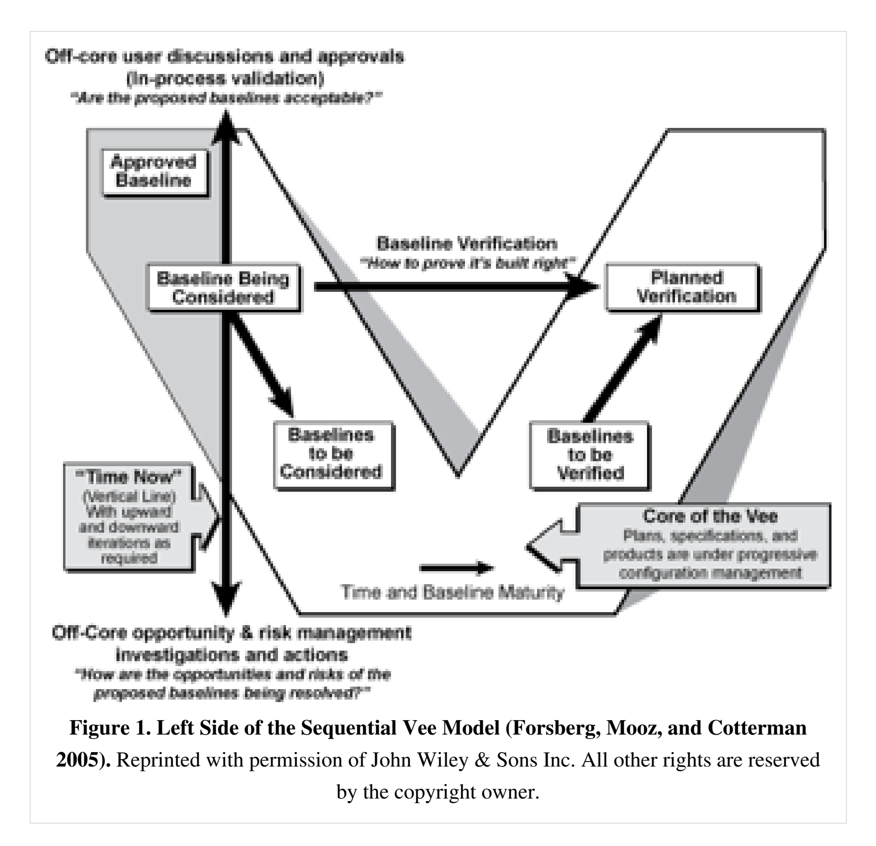


Figure 1. Left Side of the Sequential Vee Model (Forsberg, Mooz, and Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

The Vee Model endorses the INCOSE *Systems Engineering Handbook* (INCOSE 2012) definition of life cycle stages and their purposes or activities, as shown in Figure 2 below.

Life Cycle Stages

Stage	Purposes
Exploratory	Investigate new opportunities Explore technology readiness Evaluate pre-concept match with users' needs
Concept	Identify stakeholders needs Evaluate alternate concepts Recommend possible solutions
Development	Develop detailed planning Identify and manage risks and business opportunities Perform IV & V activities
Production	Produce systems Inspect and Test
Utilization	Operate system to satisfy users' needs
Support	Provide sustained system capability
Retirement	Store, archive or dispose of system

Decision Gate (between each stage)

ALTERNATIVE DECISION GATE OUTCOMES

- Proceed to next stage
- Proceed but open action items must be resolved
- Not ready; repeat the previous stage
- Terminate the project

Figure 2. An Example of Stages, Their Purposes and Major Decision Gates. (SEBoK Original)

The INCOSE *Systems Engineering Handbook* 3.2.2 contains a more detailed version of the Vee diagram (2012, Figures 3-4, p. 27) which incorporates life cycle activities into the more generic Vee model. A similar diagram, developed at the U.S. Defense Acquisition University (DAU), can be seen in Figure 3 below.

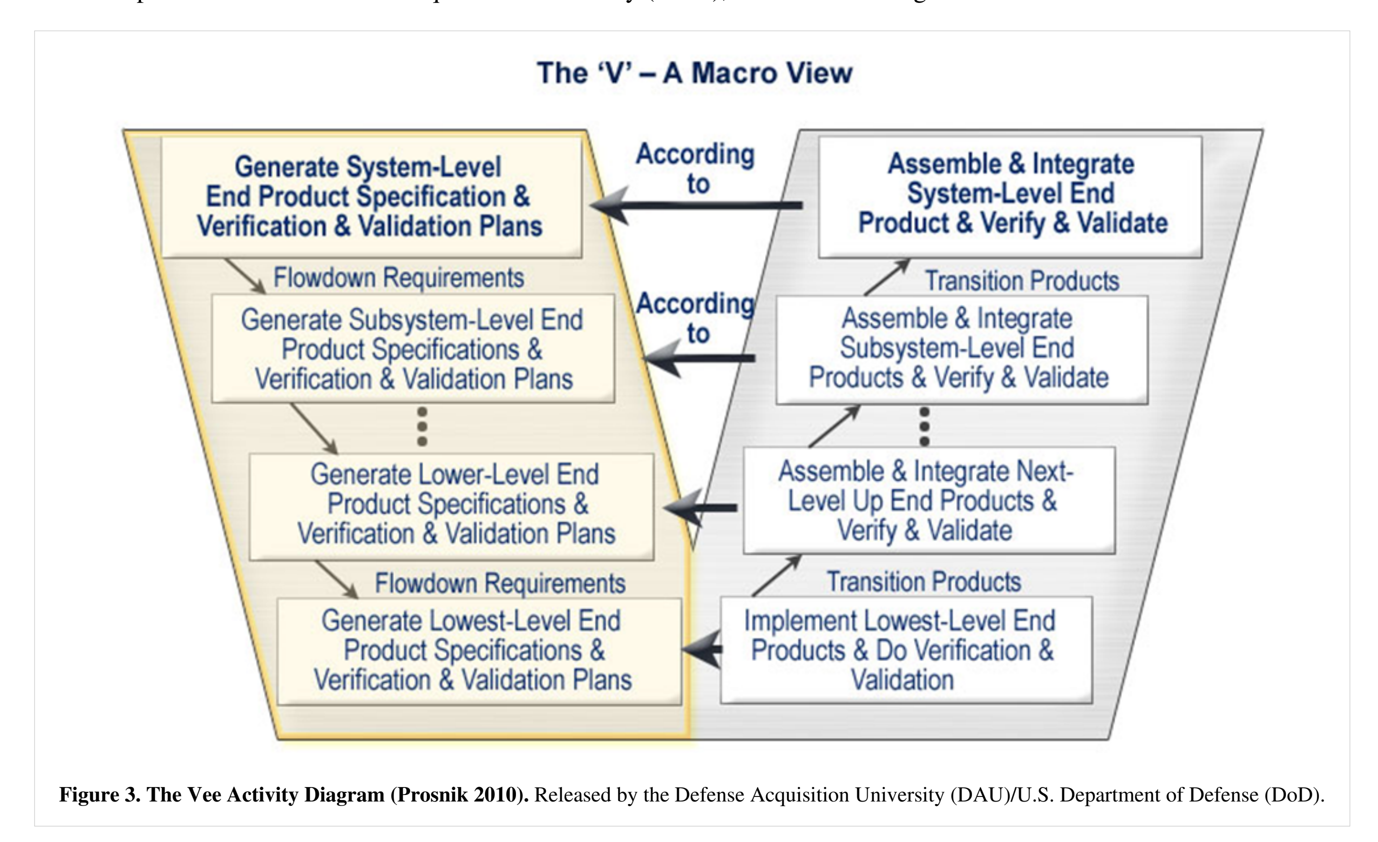


Figure 3. The Vee Activity Diagram (Prosnik 2010). Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Application of the Vee Model

Lawson (Lawson 2010) elaborates on the activities in each life cycle stage and notes that it is useful to consider the structure of a generic life cycle stage model for any type of system-of-interest (SoI) as portrayed in Figure 4. This **(T)** model indicates that one or more definition stages precede a production stage(s) where the implementation (acquisition, provisioning, or development) of two or more system elements has been accomplished.

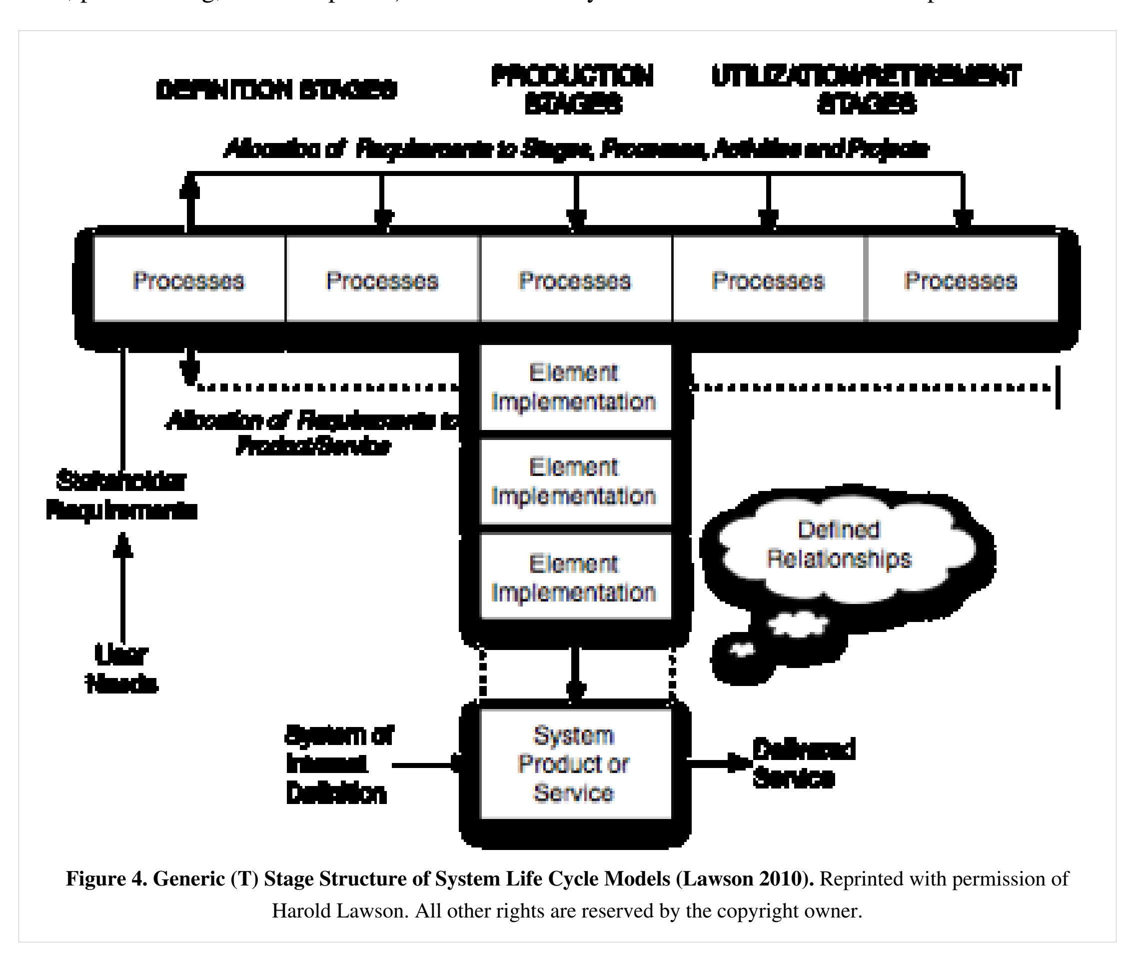


Figure 4. Generic (T) Stage Structure of System Life Cycle Models (Lawson 2010). Reprinted with permission of Harold Lawson. All other rights are reserved by the copyright owner.

Figure 5 shows the generic life cycle stages for a variety of stakeholders, from a standards organization (ISO/IEC) to commercial and government organizations. Although these stages differ in detail, they all have a similar sequential format that emphasizes the core activities as noted in Figure 2 (definition, production, and utilization/retirement).

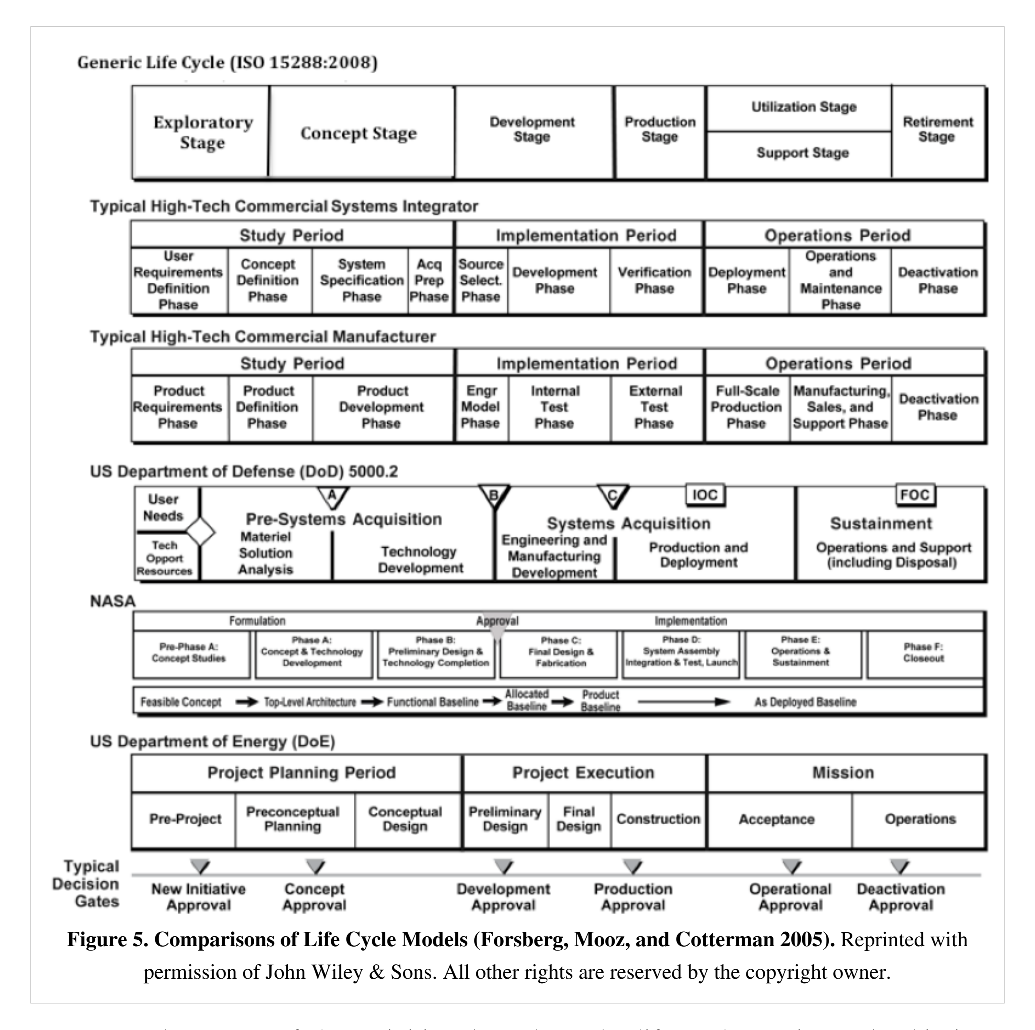


Figure 5. Comparisons of Life Cycle Models (Forsberg, Mooz, and Cotterman 2005). Reprinted with permission of John Wiley & Sons. All other rights are reserved by the copyright owner.

It is important to note that many of the activities throughout the life cycle are iterated. This is an example of recursion (glossary) as discussed in the Part 3 Introduction.

Fundamentals of Life Cycle Stages and Program Management Phase

For this discussion, it is important to note that

- The term **stage** refers to the different states of a system during its life cycle; some stages may overlap in time, such as the utilization stage and the support stage. The term “stage” is used in ISO/IEC/IEEE 15288.
- The term **phase** refers to the different steps of the program that support and manage the life of the system; the phases usually do not overlap. The term “phase” is used in many well-established models as an equivalent to the term “stage.”

Program management employs phases, milestones, and decision gates which are used to assess the evolution of a system through its various stages. The stages contain the activities performed to achieve goals and serve to control and manage the sequence of stages and the transitions between each stage. For each project, it is essential to define and publish the terms and related definitions used on respective projects to minimize confusion.

A typical program is composed of the following phases:

- The **pre-study phase**, which identifies potential opportunities to address user needs with new solutions that make business sense.
- The **feasibility phase** consists of studying the feasibility of alternative concepts to reach a second decision gate before initiating the execution stage. During the feasibility phase, stakeholders' requirements and system requirements are identified, viable solutions are identified and studied, and virtual prototypes (glossary) can be

implemented. During this phase, the decision to move forward is based on

  - whether a concept is feasible and is considered able to counter an identified threat or exploit an opportunity;
  - whether a concept is sufficiently mature to warrant continued development of a new product or line of products; and
  - whether to approve a proposal generated in response to a request for proposal.
- The **execution phase** includes activities related to four stages of the system life cycle: *development, production, utilization, and support*. Typically, there are two decision gates and two milestones associated with execution activities. The first milestone provides the opportunity for management to review the plans for execution before giving the go-ahead. The second milestone provides the opportunity to review progress before the decision is made to initiate production. The decision gates during execution can be used to determine whether to produce the developed SoI and whether to improve it or retire it.

These program management views apply not only to the SoI, but also to its elements and structure.

Life Cycle Stages

Variations of the Vee model deal with the same general stages of a life cycle:

- New projects typically begin with an exploratory research phase which generally includes the activities of concept definition, specifically the topics of business or mission analysis and the understanding of stakeholder needs and requirements. These mature as the project goes from the exploratory stage to the concept stage to the development stage.
- The production phase includes the activities of system definition and system realization, as well as the development of the system requirements (glossary) and architecture (glossary) through verification and validation.
- The utilization phase includes the activities of system deployment and system operation.
- The support phase includes the activities of system maintenance, logistics, and product and service life management, which may include activities such as service life extension or capability updates, upgrades, and modernization.
- The retirement phase includes the activities of disposal and retirement, though in some models, activities such as service life extension or capability updates, upgrades, and modernization are grouped into the "retirement" phase.

Additional information on each of these stages can be found in the sections below (see links to additional Part 3 articles above for further detail). It is important to note that these life cycle stages, and the activities in each stage, are supported by a set of systems engineering management processes.

Exploratory Research Stage

User requirements analysis and agreement is part of the exploratory research stage and is critical to the development of successful systems. Without proper understanding of the user needs, any system runs the risk of being built to solve the wrong problems. The first step in the exploratory research phase is to define the user (and stakeholder) requirements and constraints. A key part of this process is to establish the feasibility of meeting the user requirements, including technology readiness assessment. As with many SE activities this is often done iteratively, and stakeholder needs and requirements are revisited as new information becomes available.

A recent study by the National Research Council (National Research Council 2008) focused on reducing the development time for US Air Force projects. The report notes that, “simply stated, systems engineering is the translation of a user’s needs into a definition of a system and its architecture through an iterative process that results in an effective system design.” The iterative involvement with stakeholders is critical to the project success.

Except for the first and last decision gates of a project, the gates are performed simultaneously. See Figure 6 below.

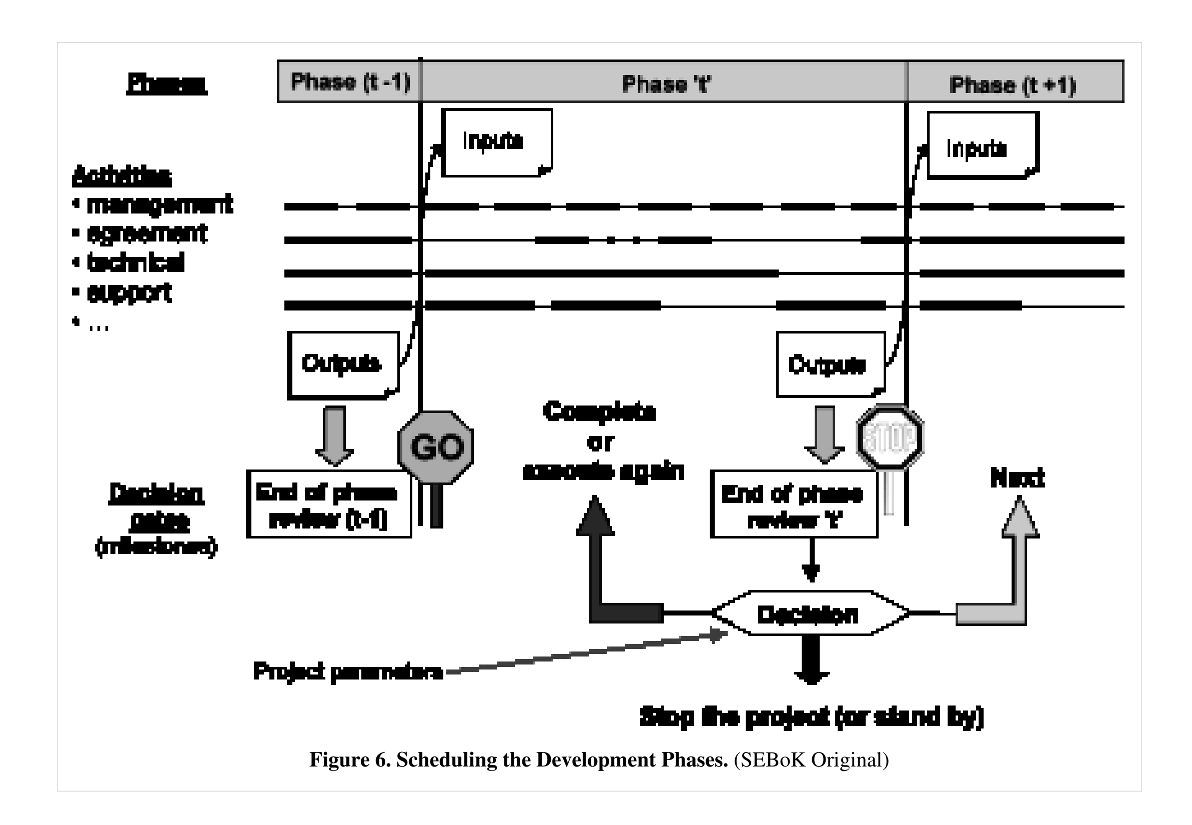


Figure 6. Scheduling the Development Phases. (SEBoK Original)

Concept Stage

During the concept stage, alternate concepts are created to determine the best approach to meet stakeholder needs. By envisioning alternatives and creating models, including appropriate prototypes, stakeholder needs will be clarified and the driving issues highlighted. This may lead to an incremental or evolutionary approach to system development. Several different concepts may be explored in parallel.

Development Stage

The selected concept(s) identified in the concept stage are elaborated in detail down to the lowest level to produce the solution that meets the stakeholder requirements. Throughout this stage, it is vital to continue with user involvement through in-process validation (the upward arrow on the Vee models). On hardware, this is done with frequent program reviews and a customer resident representative(s) (if appropriate). In agile development, the practice is to have the customer representative integrated into the development team.

Production Stage

The production stage is where the SoI is built or manufactured. Product modifications may be required to resolve production problems, to reduce production costs, or to enhance product or SoI capabilities. Any of these modifications may influence system requirements and may require system re-qualification, re-verification, or re-validation. All such changes require SE assessment before changes are approved.

Utilization Stage

A significant aspect of product life cycle management is the provisioning of supporting systems which are vital in sustaining operation of the product. While the supplied product or service may be seen as the narrow system-of-interest (NSOI) for an acquirer, the acquirer also must incorporate the supporting systems into a wider system-of-interest (WSOI). These supporting systems should be seen as system assets that, when needed, are activated in response to a situation that has emerged in respect to the operation of the NSOI. The collective name for the set of supporting systems is the integrated logistics support (ILS) system.

It is vital to have a holistic view when defining, producing, and operating system products and services. In Figure 7, the relationship between system design and development and the ILS requirements is portrayed.

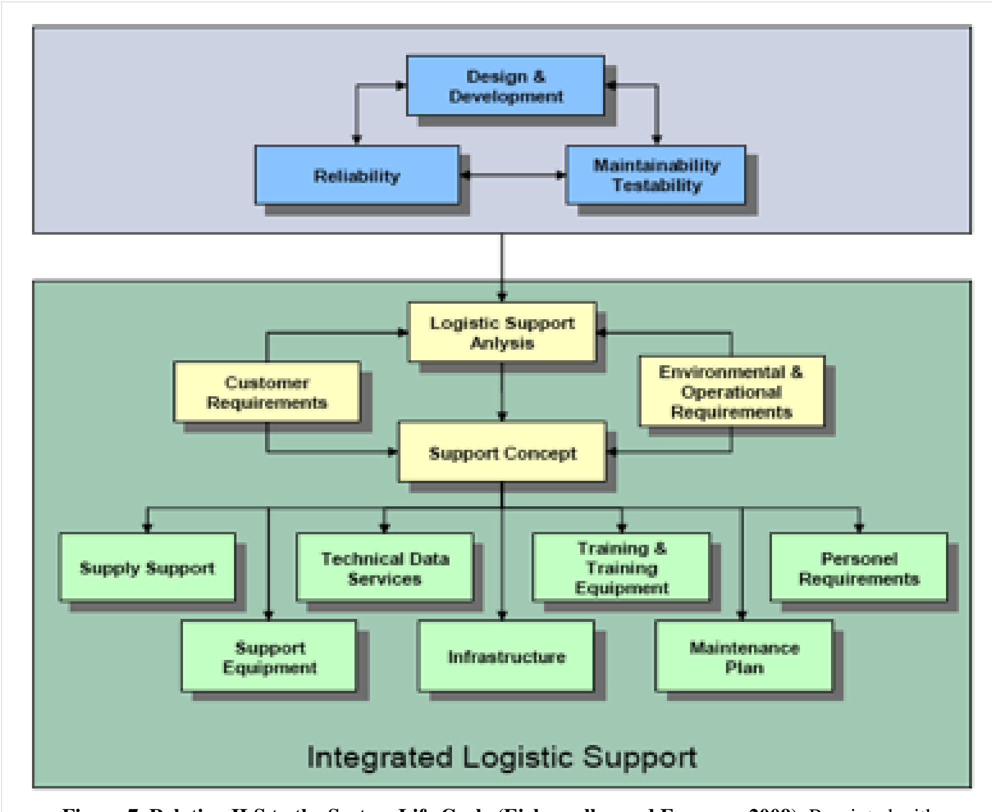


Figure 7. Relating ILS to the System Life Cycle (Eichmueller and Foreman 2009). Reprinted with permission of of ASD/AIA S3000L Steering Committee. All other rights are reserved by the copyright owner.

The requirements for reliability, resulting in the need of maintainability and testability, are driving factors.

Support Stage

In the support stage, the SoI is provided services that enable continued operation. Modifications may be proposed to resolve supportability problems, to reduce operational costs, or to extend the life of a system. These changes require SE assessment to avoid loss of system capabilities while under operation. The corresponding technical process is the maintenance process.

Retirement Stage

In the retirement stage, the SoI and its related services are removed from operation. SE activities in this stage are primarily focused on ensuring that disposal requirements are satisfied. In fact, planning for disposal is part of the system definition during the concept stage. Experiences in the 20th century repeatedly demonstrated the consequences when system retirement and disposal was not considered from the outset. Early in the 21st century, many countries have changed their laws to hold the creator of a SoI accountable for proper end-of-life disposal of the system.

Life Cycle Reviews

To control the progress of a project, different types of reviews are planned. The most commonly used are listed as follows, although the names are not universal:

- The **system requirements review** (SRR) is planned to verify and validate the set of system requirements before starting the detailed design activities.
- The preliminary design review (PDR) is planned to verify and validate the set of system requirements, the design artifacts, and justification elements at the end of the first engineering loop (also known as the "design-to" gate).
- The critical design review (CDR) is planned to verify and validate the set of system requirements, the design artifacts, and justification elements at the end of the last engineering loop (the “build-to” and “code-to” designs are released after this review).
- The integration, verification, and validation reviews are planned as the components are assembled into higher level subsystems and elements. A sequence of reviews is held to ensure that everything integrates properly and that there is objective evidence that all requirements have been met. There should also be an in-process validation that the system, as it is evolving, will meet the stakeholders’ requirements (see Figure 7).
- The final validation review is carried out at the end of the integration phase.
- Other management related reviews can be planned and conducted in order to control the correct progress of work, based on the type of system and the associated risks.

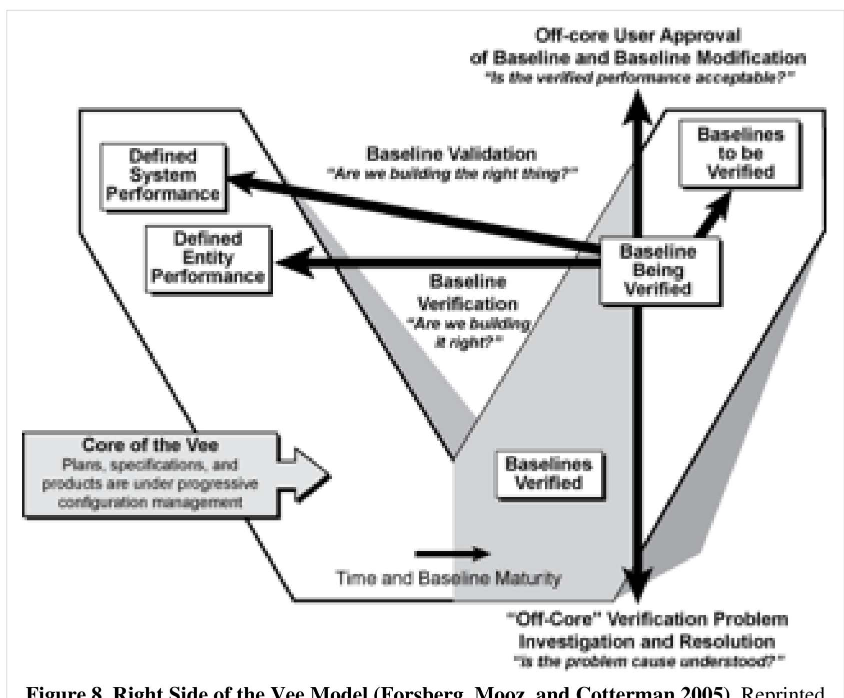


Figure 8. Right Side of the Vee Model (Forsberg, Mooz, and Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

References

Works Cited

Eichmueller, P. and B. Foreman. 2010. *S3000LTM*. Brussels, Belgium: Aerospace and Defence Industries Association of Europe (ASD)/Aerospace Industries Association (AIA).

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications, Kings College, UK.

Primary References

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto". in *The Agile Manifesto* [database online]. Accessed December 04 2014 at www.agilemanifesto.org/principles.html

Boehm, B. and R. Turner. 2004. *Balancing Agility and Discipline.* New York, NY, USA: Addison-Wesley.

Fairley, R. 2009. *Managing and Leading Software Projects.* New York, NY, USA: J. Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* 3rd ed. New York, NY, USA: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* Kings College, UK: College Publications.

Pew, R., and A. Mavor (eds.) 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Royce, W.E. 1998. *Software Project Management: A Unified Framework*. New York, NY, USA: Addison Wesley.

Additional References

Anderson, D. 2010. *Kanban*. Sequim, WA, USA: Blue Hole Press.

Baldwin, C. and K. Clark. 2000. *Design Rules: The Power of Modularity.* Cambridge, MA, USA: MIT Press.

Beck, K. 1999. *Extreme Programming Explained.* New York, NY, USA: Addison Wesley.

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto". in The Agile Manifesto [database online]. Accessed 2010. Available at: www.agilemanifesto.org/principles.html

Biffl, S., A. Aurum, B. Boehm, H. Erdogmus, and P. Gruenbacher (eds.). 2005. *Value-Based Software Engineering*. New York, NY, USA: Springer.

Boehm, B. 1988. “A Spiral Model of Software Development.” IEEE *Computer* 21(5): 61-72.

Boehm, B. 2006. “Some Future Trends and Implications for Systems and Software Engineering Processes.” *Systems Engineering.* 9(1): 1-19.

Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy. 1998. “Using the WinWin Spiral Model: A Case Study.” IEEE *Computer.* 31(7): 33-44.

Boehm, B., R. Turner, J. Lane, S. Koolmanojwong. 2014 (in press). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model*. Boston, MA, USA: Addison Wesley.

Castellano, D.R. 2004. "Top Five Quality Software Projects." *CrossTalk.* 17(7) (July 2004): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2004/200407/200407-0-Issue.pdf

Checkland, P. 1981. *Systems Thinking, Systems Practice*. New York, NY, USA: Wiley.

Crosson, S. and B. Boehm. 2009. "Adjusting Software Life cycle Anchorpoints: Lessons Learned in a System of Systems Context." Proceedings of the Systems and Software Technology Conference, 20-23 April 2009, Salt Lake City, UT, USA.

Dingsoyr, T., T. Dyba. and N. Moe (eds.). 2010. "Agile Software Development: Current Research and Future Directions." Chapter in B. Boehm, J. Lane, S. Koolmanjwong, and R. Turner, *Architected Agile Solutions for Software-Reliant Systems.* New York, NY, USA: Springer.

Dorner, D. 1996. *The Logic of Failure*. New York, NY, USA: Basic Books.

Forsberg, K. 1995. "'If I Could Do That, Then I Could…' System Engineering in a Research and Development Environment." Proceedings of the Fifth Annual International Council on Systems Engineering (INCOSE) International Symposium. 22-26 July 1995. St. Louis, MO, USA.

Forsberg, K. 2010. "Projects Don't Begin With Requirements." Proceedings of the IEEE Systems Conference, 5-8 April 2010, San Diego, CA, USA.

Gilb, T. 2005. *Competitive Engineering*. Maryland Heights, MO, USA: Elsevier Butterworth Heinemann.

Goldratt, E. 1984. *The Goal*. Great Barrington, MA, USA: North River Press.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* New York, NY, USA: Wiley.

Holland, J. 1998. *Emergence*. New York, NY, USA: Perseus Books.

ISO/IEC. 2010. Systems and Software Engineering, Part 1: Guide for Life Cycle Management. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 24748-1:2010.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes.* Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / , Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC. 2003. *Systems Engineering — A Guide for The Application of ISO/IEC 15288 System Life Cycle Processes*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 19760:2003 (E).

Jarzombek, J. 2003. "Top Five Quality Software Projects." *CrossTalk.* 16(7) (July 2003): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2003/200307/200307-0-Issue.pdf.

Kruchten, P. 1999. *The Rational Unified Process.* New York, NY, USA: Addison Wesley.

Landis, T. R. 2010. *Lockheed Blackbird Family (A-12, YF-12, D-21/M-21 & SR-71).* North Branch, MN, USA: Specialty Press.

Madachy, R. 2008. *Software Process Dynamics*. Hoboken, NJ, USA: Wiley.

Maranzano, J.F., S.A. Rozsypal, G.H. Zimmerman, G.W. Warnken, P.E. Wirth, D.W. Weiss. 2005. "Architecture Reviews: Practice and Experience." IEEE *Software.* 22(2): 34-43.

National Research Council of the National Academies (USA). 2008. *Pre-Milestone A and Early-Phase Systems Engineering*. Washington, DC, USA: The National Academies Press.

Osterweil, L. 1987. "Software Processes are Software Too." Proceedings of the SEFM 2011: 9th International Conference on Software Engineering. Monterey, CA, USA.

Poppendeick, M. and T. Poppendeick. 2003. *Lean Software Development: an Agile Toolkit.* New York, NY, USA: Addison Wesley.

Rechtin, E. 1991. *System Architecting: Creating and Building Complex Systems.* Upper Saddle River, NY, USA: Prentice-Hall.

Rechtin, E., and M. Maier. 1997. *The Art of System Architecting*. Boca Raton, FL, USA: CRC Press.

Schwaber, K. and M. Beedle. 2002. *Agile Software Development with Scrum*. Upper Saddle River, NY, USA: Prentice Hall.

Spruill, N. 2002. "Top Five Quality Software Projects." *CrossTalk.* 15(1) (January 2002): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2002/200201/200201-0-Issue.pdf.

Stauder, T. 2005. "Top Five Department of Defense Program Awards." *CrossTalk.* 18(9) (September 2005): 4-13. Available at http://www.crosstalkonline.org/storage/issue-archives/2005/200509/200509-0-Issue.pdf.

Warfield, J. 1976. *Societal Systems: Planning, Policy, and Complexity*. New York, NY, USA: Wiley.

Womack, J. and D. Jones. 1996. *Lean Thinking.* New York, NY, USA: Simon and Schuster.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Life Cycle Process Models: Iterative

There are a large number of life cycle process models. As discussed in the System Life Cycle Process Drivers and Choices article, these models fall into three major categories: (1) primarily pre-specified and sequential processes; (2) primarily evolutionary and concurrent processes (e.g., the rational unified process and various forms of the Vee and spiral models); and (3) primarily interpersonal and unconstrained processes (e.g., agile development, Scrum, extreme programming (XP), dynamic system development methods, and innovation-based processes).

This article discusses incremental and evolutionary development models (the second and third categories listed above) beyond variants of the Vee model. While there are a number of different models describing the project environment, the spiral model and the Vee Model have become the dominant approaches to visualizing the development process. Both the Vee and the spiral are useful models that emphasize different aspects of a system life cycle.

General implications of using iterative models for system design and development are discussed below. For a more specific understanding of how this life cycle model impacts systems engineering activities, please see the other knowledge areas (KAs) in Part 3. This article is focused on the use of iterative life cycle process models in systems engineering; however, because iterative process models are commonly used in software development, many of the examples below come from software projects. (See Systems Engineering and Software Engineering in Part 6 for more information on life cycle implications in software engineering.)

Incremental and Evolutionary Development

Overview of the Incremental Approach

Incremental and iterative development (IID) methods have been in use since the 1960s (and perhaps earlier). They allow a project to provide an initial capability followed by successive deliveries to reach the desired system-of-interest (SoI).

The IID approach, shown in Figure 1, is used when

- rapid exploration and implementation of part of the system is desired;
- the requirements are unclear from the beginning;
- funding is constrained;
- the customer wishes to hold the SoI open to the possibility of inserting new technology at a later time; and/or
- experimentation is required to develop successive prototype (glossary) versions.

The attributes that distinguish IID from the single-pass, plan-driven approach are velocity and adaptability.

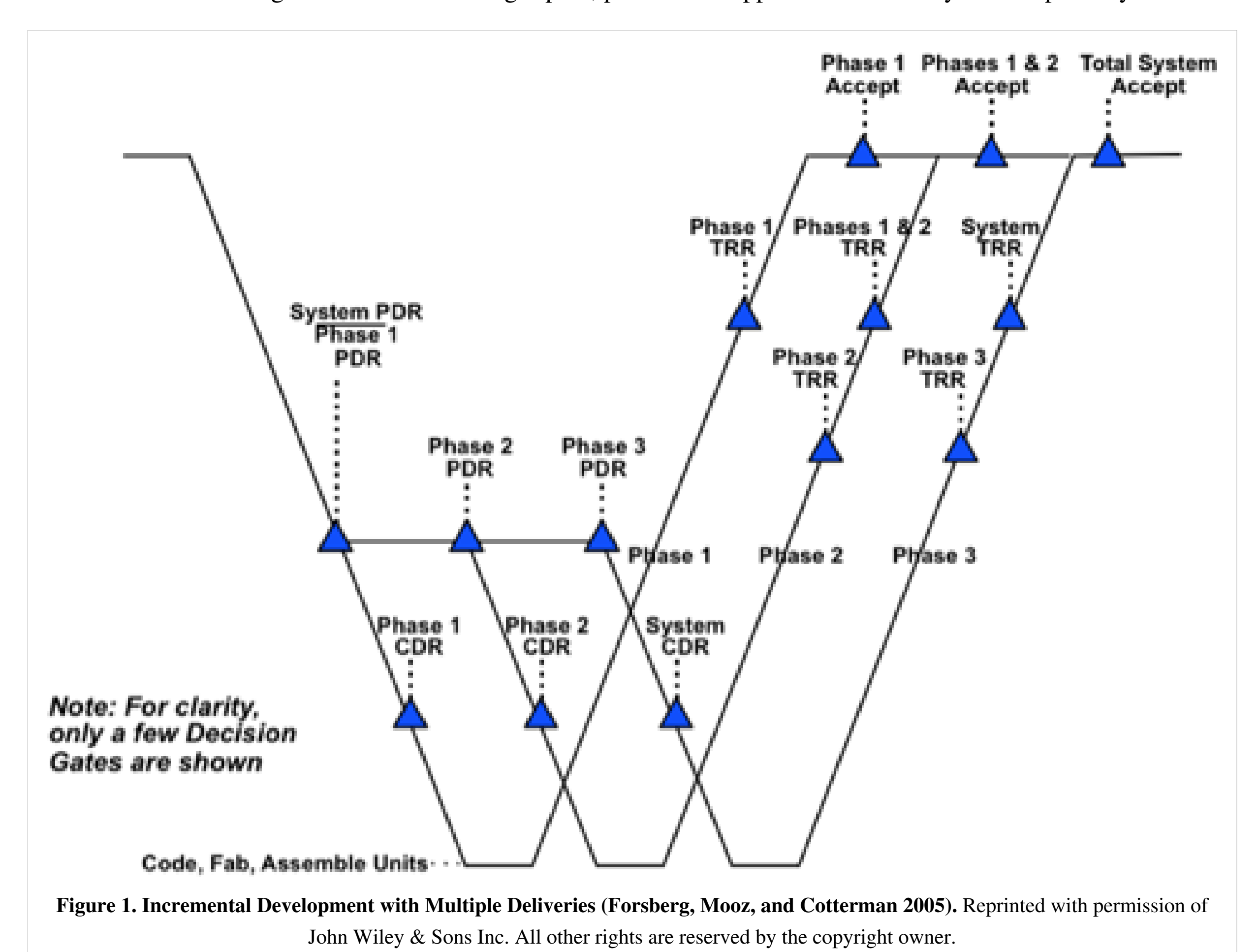


Figure 1. Incremental Development with Multiple Deliveries (Forsberg, Mooz, and Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Incremental development may also be “plan-driven” in nature if the requirements are known early on in the life cycle. The development of the functionality is performed incrementally to allow for insertion of the latest technology or for potential changes in needs or requirements. IID also imposes constraints. The example shown in Figure 2 uses the increments to develop high-risk subsystems (or components) early, but the system cannot function until all increments are complete.

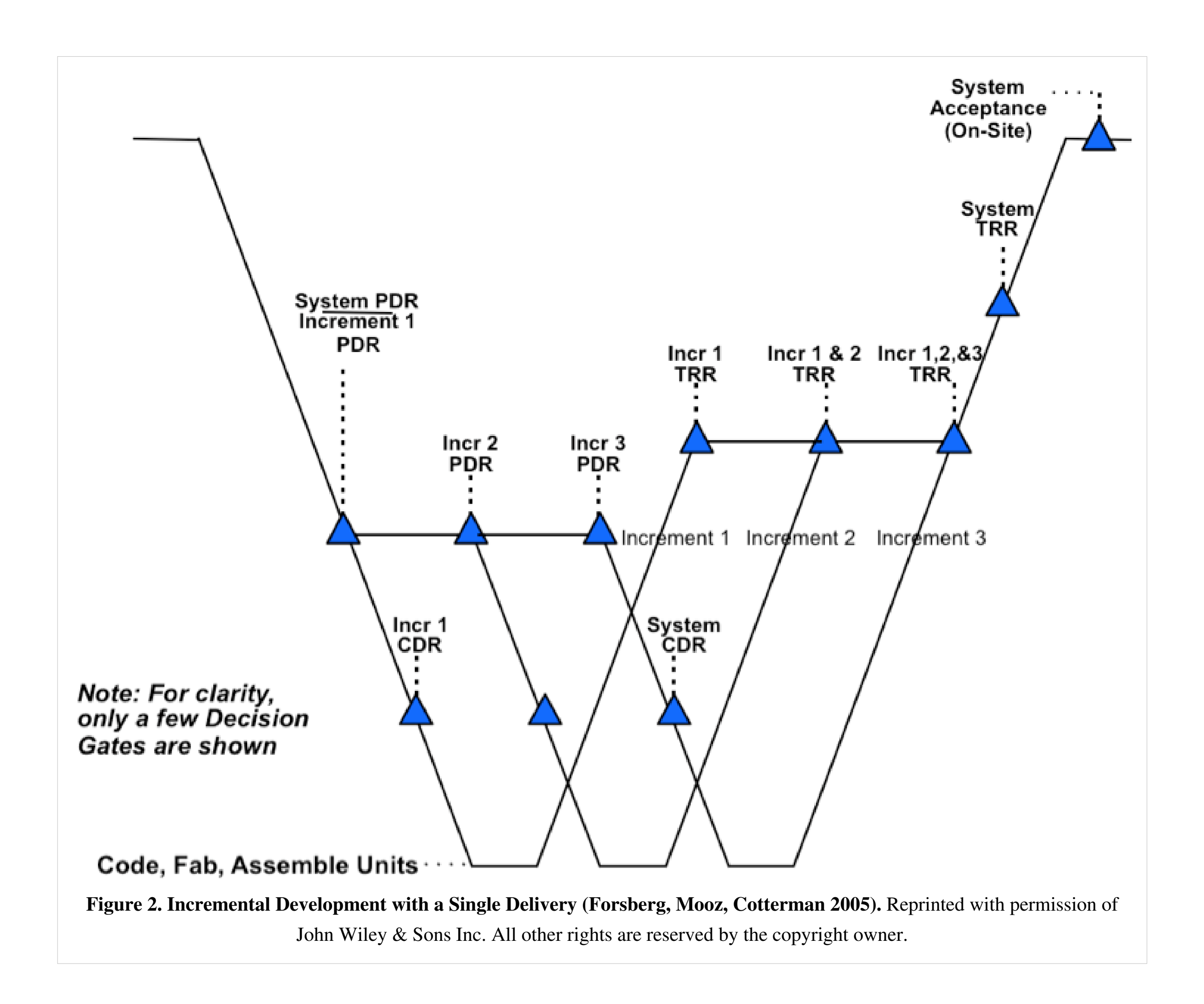


Figure 2. Incremental Development with a Single Delivery (Forsberg, Mooz, Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Overview of the Evolutionary Approach

A specific IID methodology called evolutionary development is common in research and development (R&D) environments in both the government and commercial sector. Figure 3 illustrates this approach, which was used in the evolution of the high temperature tiles for the NASA Space Shuttle (Forsberg 1995). In the evolutionary approach, the end state of each phase of development is unknown, though the goal is for each phase to result in some sort of useful product.

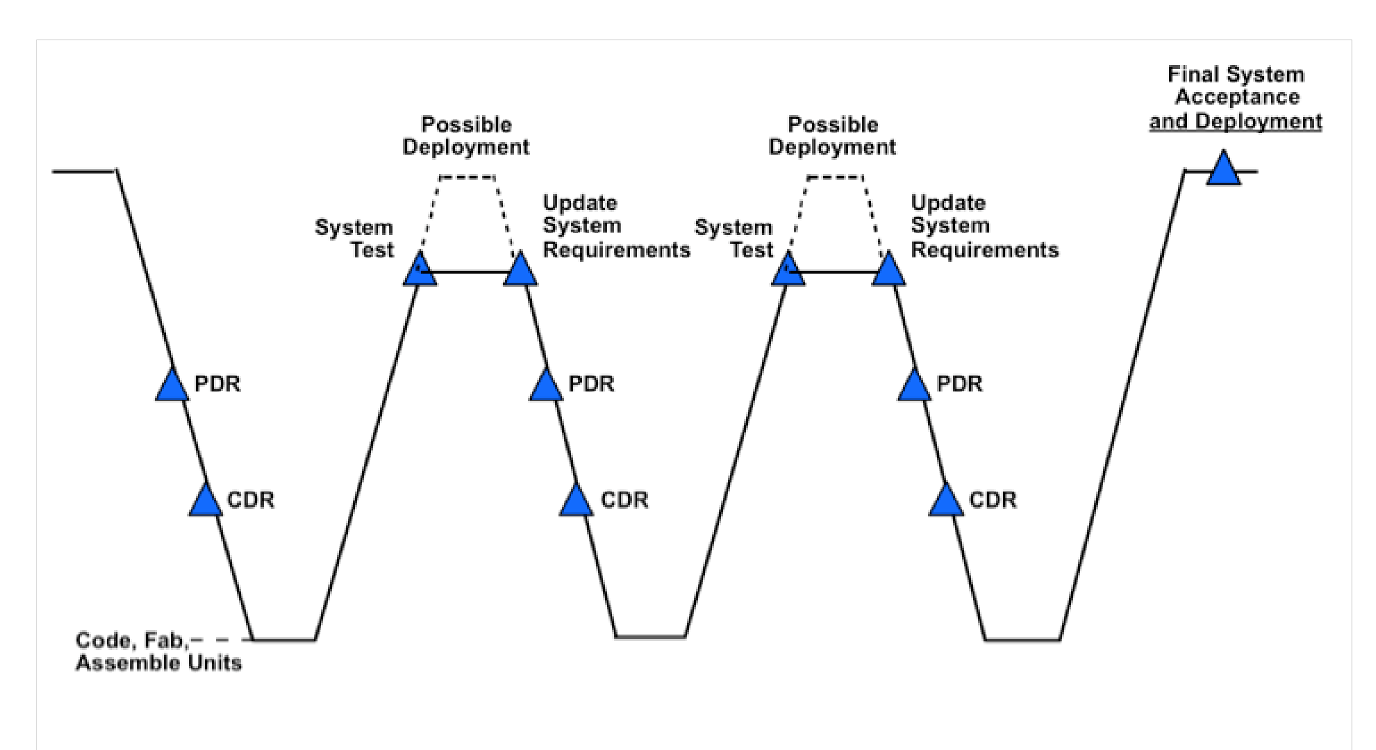


Figure 3. Evolutionary Generic Model (Forsberg, Mooz, Cotterman 2005). Reprinted with permission of John Wiley & Sons, Inc.

The real world development environment is complex and difficult to map because many different project cycles are simultaneously underway. Figure 4 shows the applied research era for the development of the space shuttle Orbiter and illustrates multi-levels of simultaneous development, trade-studies, and ultimately, implementation.

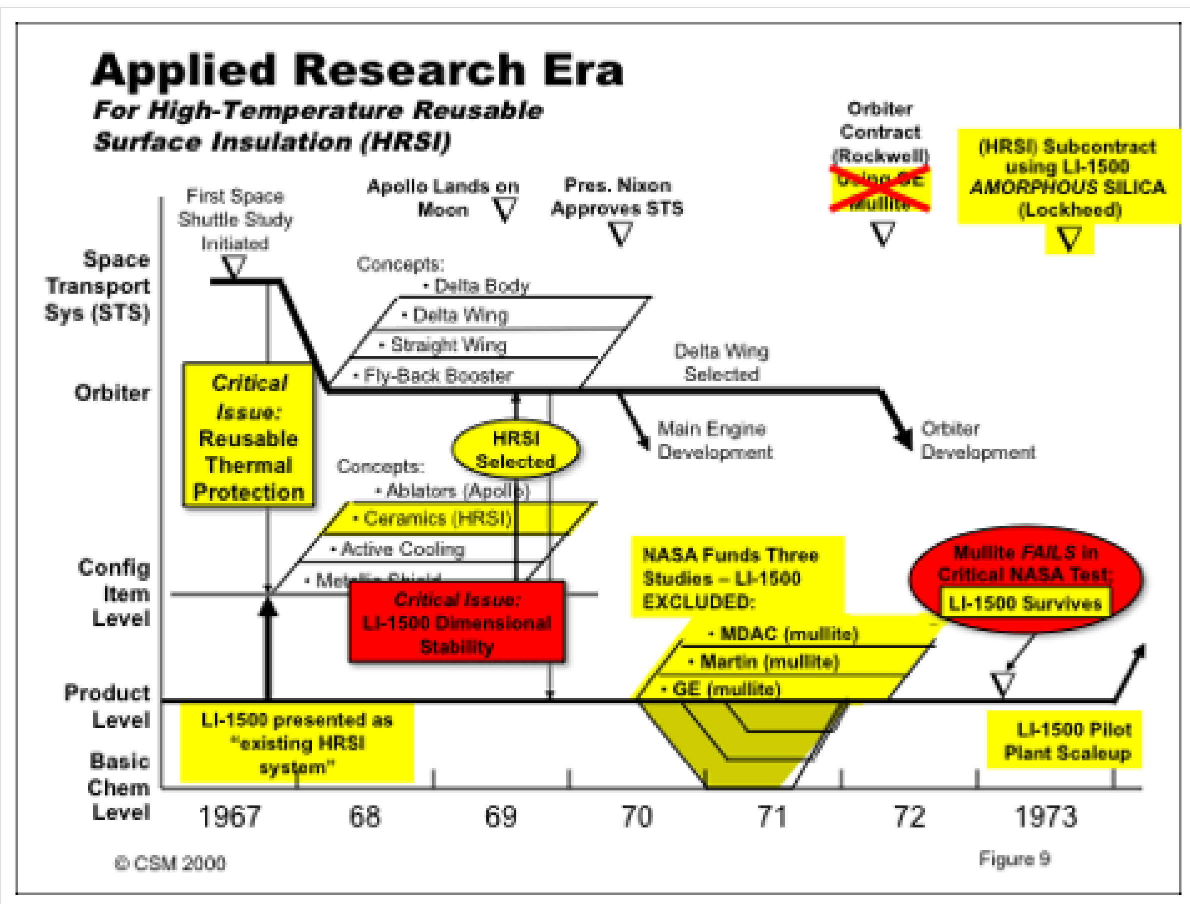


Figure 4. Evolution of Components and Orbiter Subsystems (including space shuttle tiles) During Creation of a Large "Single-Pass" Project (Forsberg 1995). Reprinted with permission of Kevin Forsberg. All other rights are reserved by the copyright owner.

Iterative Software Development Process Models

Software is a flexible and malleable medium which facilitates iterative analysis, design, construction, verification, and validation to a greater degree than is usually possible for the purely physical components of a system. Each repetition of an iterative development model adds material (code) to the growing software base; the expanded code base is tested, reworked as necessary, and demonstrated to satisfy the requirements for the baseline.

Process models for software development support iterative development on cycles of various lengths. Table 1 lists three iterative software development models which are presented in more detail below, as well as the aspects of software development that are emphasized by those models.

Table 1. Primary Emphases of Three Iterative Software Development Models.

Iterative Model	Emphasis
Incremental-build	Iterative implementation-verification-validations-demonstration cycles
Spiral	Iterative risk-based analysis of alternative approaches and evaluation of outcomes
Agile	Iterative evolution of requirements and code

Please note that the information below is focused specifically on the utilization of different life cycle models for software systems. In order to better understand the interactions between software engineering (SwE) and systems engineering (SE), please see the Systems Engineering and Software Engineering KA in Part 6.

Overview of Iterative-Development Process Models

Developing and modifying software involves creative processes that are subject to many external and changeable forces. Long experience has shown that it is impossible to “get it right” the first time, and that iterative development processes are preferable to linear, sequential development process models, such as the well-known Waterfall model. In iterative development, each cycle of the iteration subsumes the software of the previous iteration and adds new capabilities to the evolving product to create an expanded version of the software. Iterative development processes provide the following advantages:

- Continuous integration, verification, and validation of the evolving product;
- Frequent demonstrations of progress;
- Early detection of defects;
- Early warning of process problems;
- Systematic incorporation of the inevitable rework that occurs in software development; and
- Early delivery of subset capabilities (if desired).

Iterative development takes many forms in SwE, including the following:

- An incremental-build process, which is used to produce periodic (typically weekly) builds of increasing product capabilities;
- Agile development, which is used to closely involve a prototypical customer in an iterative process that may repeat on a daily basis; and
- The spiral model, which is used to confront and mitigate risk factors encountered in developing the successive versions of a product.

The Incremental-Build Model

The incremental-build model is a build-test-demonstrated model of iterative cycles in which frequent demonstrations of progress, verification, and validation of work-to-date are emphasized. The model is based on stable requirements and a software architectural specification. Each build adds new capabilities to the incrementally growing product. The process ends when the final version is verified, validated, demonstrated, and accepted by the customer.

Table 2 lists some partitioning criteria for incremental development into incremental build units of (typically) one calendar week each. The increments and the number of developers available to work on the project determine the number of features that can be included in each incremental build. This, in turn, determines the overall schedule.

Table 2. Some partitioning criteria for incremental builds (Fairley 2009). Reprinted with permission of the IEEE Computer Society and John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Kind of System	Partitioning Criteria
Application package	Priority of features
Safety-critical systems	Safety features first; prioritized others follow
User-intensive systems	User interface first; prioritized others follow
System software	Kernel first; prioritized utilities follow

Figure 5 illustrates the details of the build-verify-validate-demonstrate cycles in the incremental build process. Each build includes detailed design, coding, integration, review, and testing done by the developers. In cases where code is to be reused without modification, some or all of an incremental build may consist of review, integration, and testing of the base code augmented with the reused code. It is important to note that development of an increment may result in reworking previous components developed for integration to fix defects.

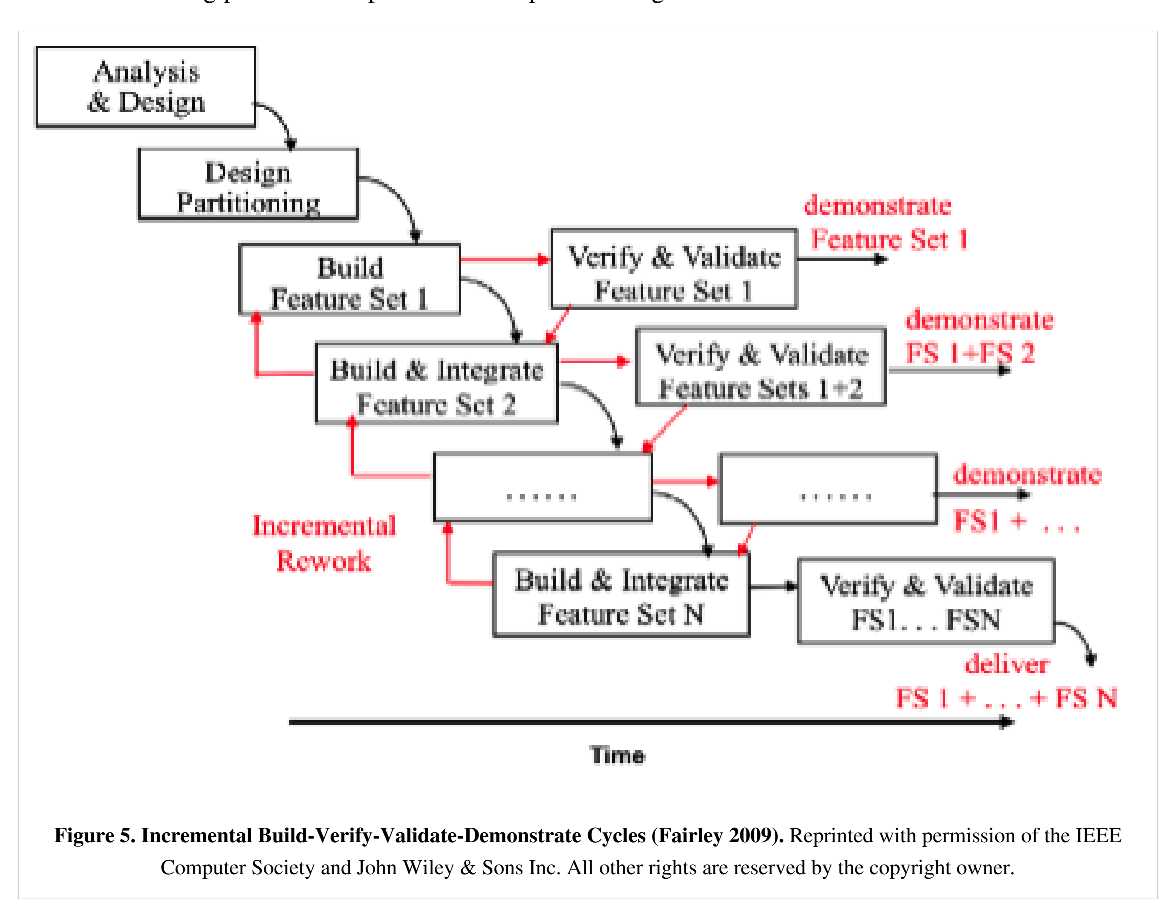


Figure 5. Incremental Build-Verify-Validate-Demonstrate Cycles (Fairley 2009). Reprinted with permission of the IEEE Computer Society and John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Incremental verification, validation, and demonstration, as illustrated in Figure 5, overcome two of the major problems of a waterfall approach by

- exposing problems early so they can be corrected as they occur; and
- incorporating minor in-scope changes to requirements that occur as a result of incremental demonstrations in subsequent builds.

Figure 5 also illustrates that it may be possible to overlap successive builds of the product. It may be possible, for example, to start a detailed design of the next version while the present version is being validated.

Three factors determine the degree of overlap that can be achieved:

1. Availability of personnel;
2. Adequate progress on the previous version; and
3. The risk of significant rework on the next overlapped build because of changes to the previous in-progress build.

The incremental build process generally works well with small teams, but can be scaled up for larger projects.

A significant advantage of an incremental build process is that features built first are verified, validated, and demonstrated most frequently because subsequent builds incorporate the features of the earlier iterations. In building the software to control a nuclear reactor, for example, the emergency shutdown software could be built first, as it would then be verified and validated in conjunction with the features of each successive build.

In summary, the incremental build model, like all iterative models, provides the advantages of continuous integration and validation of the evolving product, frequent demonstrations of progress, early warning of problems, early delivery of subset capabilities, and systematic incorporation of the inevitable rework that occurs in software development.

The Role of Prototyping in Software Development

In SwE, a prototype is a mock-up of the desired functionality of some part of the system. This is in contrast to physical systems, where a prototype is usually the first fully functional version of a system (Fairley 2009, 74).

In the past, incorporating prototype software into production systems has created many problems. Prototyping is a useful technique that should be employed as appropriate; however, prototyping is *not* a process model for software development. When building a software prototype, the knowledge gained through the development of the prototype is beneficial to the program; however, the prototype code may not be used in the deliverable version of the system. In many cases, it is more efficient and more effective to build the production code from scratch using the knowledge gained by prototyping than to re-engineer the existing code.

Life Cycle Sustainment of Software

Software, like all systems, requires sustainment efforts to enhance capabilities, adapt to new environments, and correct defects. The primary distinction for software is that sustainment efforts change the software; unlike physical entities, software components do not have to be replaced because of physical wear and tear. Changing the software requires re-verification and re-validation, which may involve extensive regression testing to determine that the change has the desired effect and has not altered other aspects of functionality or behavior.

Retirement of Software

Useful software is rarely retired; however, software that is useful often experiences many upgrades during its lifetime. A later version may bear little resemblance to the initial release. In some cases, software that ran in a former operational environment is executed on hardware emulators that provide a virtual machine on newer hardware. In other cases, a major enhancement may replace and rename an older version of the software, but the enhanced version provides all of the capabilities of the previous software in a compatible manner. Sometimes, however, a newer version of software may fail to provide compatibility with the older version, which necessitates other changes to a system.

Primarily Evolutionary and Concurrent Processes: The Incremental Commitment Spiral Model

Overview of the Incremental Commitment Spiral Model

A view of the Incremental Commitment Spiral Model (ICSM) is shown in Figure 6.

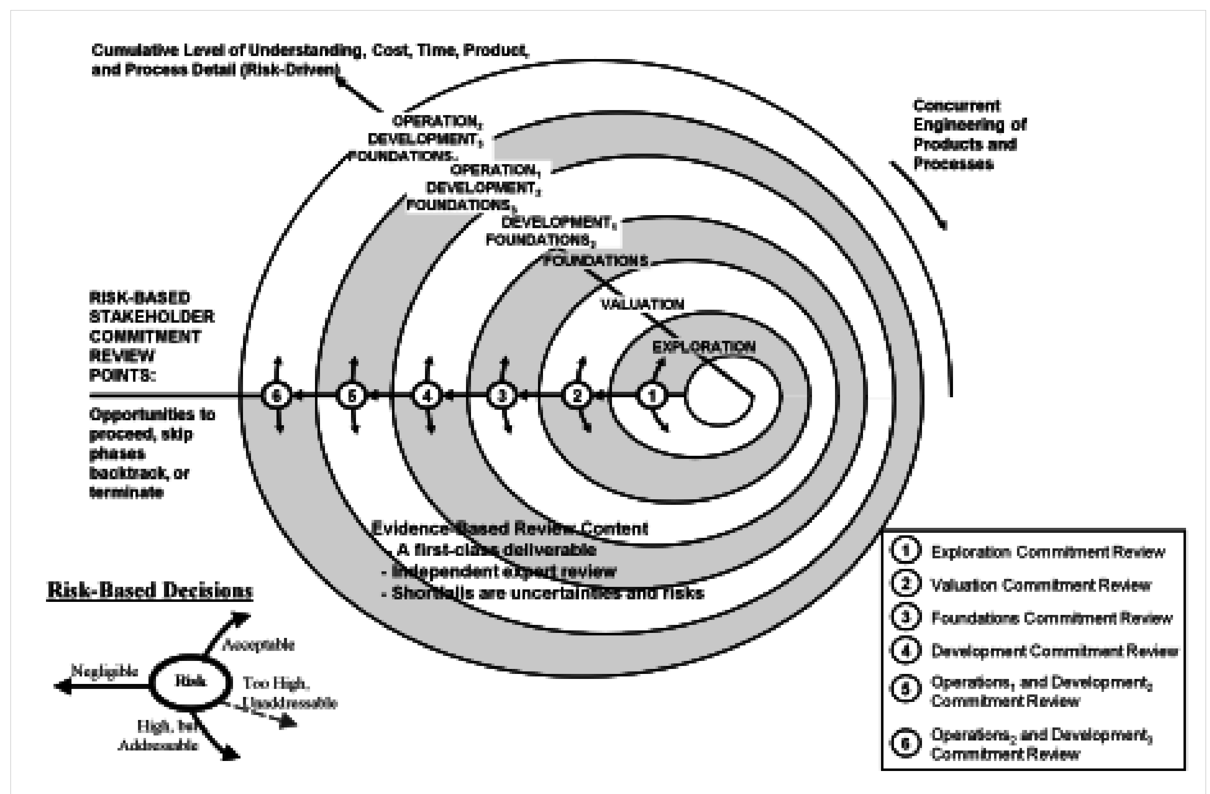


Figure 6. The Incremental Commitment Spiral Model (ICSM) (Pew and Mavor 2007). Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

In the ICSM, each spiral addresses requirements and solutions concurrently, rather than sequentially, as well as products and processes, hardware, software, and human factors aspects, and business case analyses of alternative product configurations or product line investments. The stakeholders consider the risks and risk mitigation plans and decide on a course of action. If the risks are acceptable and covered by risk mitigation plans, the project proceeds into the next spiral.

The development spirals after the first development commitment review follow the three-team incremental development approach for achieving both agility and assurance shown and discussed in Figure 2, "Evolutionary-Concurrent Rapid Change Handling and High Assurance" of System Life Cycle Process Drivers and Choices.

Other Views of the Incremental Commitment Spiral Model

Figure 7 presents an updated view of the ICSM life cycle process recommended in the National Research Council *Human-System Integration in the System Development Process* study (Pew and Mavor 2007). It was called the Incremental Commitment Model (ICM) in the study. The ICSM builds on the strengths of current process models, such as early verification and validation concepts in the Vee model, concurrency concepts in the concurrent engineering model, lighter-weight concepts in the agile and lean models, risk-driven concepts in the spiral model, the phases and anchor points in the rational unified process (RUP) (Kruchten 1999; Boehm 1996), and recent extensions of the spiral model to address systems of systems (SoS) capability acquisition (Boehm and Lane 2007).

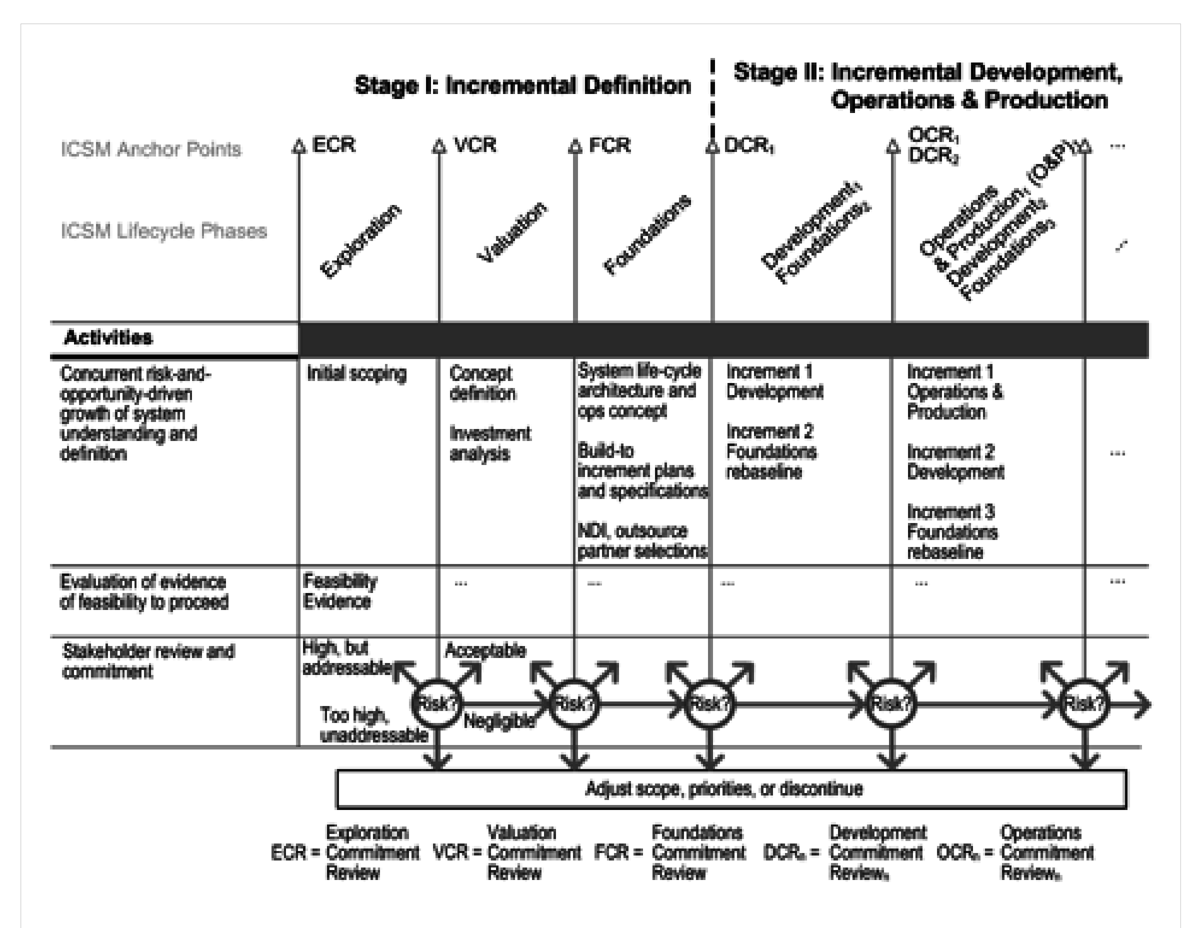


Figure 7. Phased View of the Generic Incremental Commitment Spiral Model Process (Pew and Mavor 2007). Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

The top row of activities in Figure 7 indicates that a number of system aspects are being concurrently engineered at an increasing level of understanding, definition, and development. The most significant of these aspects are shown in Figure 8, an extension of a similar *"hump diagram"* view of concurrently engineered software activities developed as part of the RUP (Kruchten 1999).

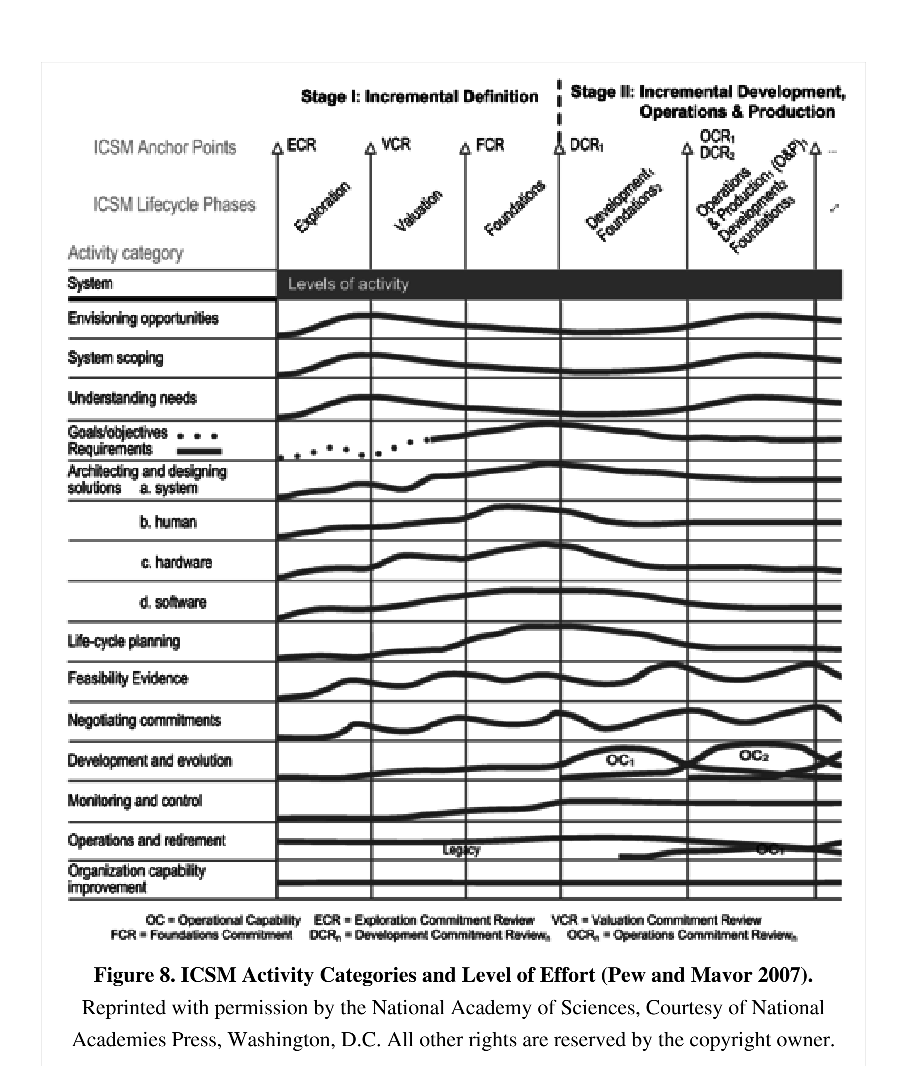


Figure 8. ICSM Activity Categories and Level of Effort (Pew and Mavor 2007).
Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

As with the RUP version, the magnitude and shape of the levels of effort will be risk-driven and likely to vary from project to project. Figure 8 indicates that a great deal of concurrent activity occurs within and across the various ICSM phases, all of which need to be *"synchronized and stabilized,"* a best-practice phrase taken from *Microsoft Secrets* (Cusumano and Selby 1996) to keep the project under control.

The review processes and use of independent experts are based on the highly successful AT&T Architecture Review Board procedures described in “Architecture Reviews: Practice and Experience” (Maranzano et al. 2005). Figure 9 shows the content of the feasibility evidence description. Showing feasibility of the concurrently-developed elements helps synchronize and stabilize the concurrent activities.

Feasibility Evidence Description Content

Evidence *provided by developer* and *validated by independent experts* that if the system is built to the specified architecture, it will:

- **Satisfy the requirements: capability, interfaces, level of service, and evolution**
- **Support the operational concept**
- **Be buildable within the budgets and schedules in the plan**
- **Generate a viable return on investment**
- **Generate satisfactory outcomes for all of the success-critical stakeholders**
- **Resolve all major risks by risk management plans**

Figure 9. Feasibility Evidence Description Content (Pew and Mavor 2007). Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

The operations commitment review (OCR) is different in that it addresses the often higher operational risks of fielding an inadequate system. In general, stakeholders will experience a two- to ten-fold increase in commitment level while going through the sequence of engineering certification review (ECR) to design certification review (DCR) milestones, but the increase in going from DCR to OCR can be much higher. These commitment levels are based on typical cost profiles across the various stages of the acquisition life cycle.

Underlying ICSM Principles

ICSM has four underlying principles which must be followed:

1. Stakeholder value-based system definition and evolution;
2. Incremental commitment and accountability;
3. Concurrent system and software definition and development; and
4. Evidence and risk-based decision making.

Model Experience to Date

The National Research Council Human-Systems Integration study (2008) found that the ICSM processes and principles correspond well with best commercial practices, as described in the Next Generation Medical Infusion Pump Case Study in Part 7. Further examples are found in *Human-System Integration in the System Development Process: A New Look* (Pew and Mavor 2007, chap. 5), *Software Project Management* (Royce 1998, Appendix D), and the annual series of "Top Five Quality Software Projects", published in CrossTalk (2002-2005).

Agile and Lean Processes

According to the INCOSE *Systems Engineering Handbook* 3.2.2, *“Project execution methods can be described on a continuum from 'adaptive' to 'predictive.' Agile methods exist on the 'adaptive' side of this continuum, which is not the same as saying that agile methods are 'unplanned' or 'undisciplined,'”* (INCOSE 2011, 179). Agile development methods can be used to support iterative life cycle models, allowing flexibility over a linear process that better aligns with the planned life cycle for a system. They primarily emphasize the development and use of tacit interpersonal knowledge as compared to explicit documented knowledge, as evidenced in the four value propositions in the **"Agile Manifesto"**:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value*

- ***Individuals and interactions*** *over processes and tools;*

- ***Working software*** *over comprehensive documentation;*
- ***Customer collaboration*** *over contract negotiation; and*
- ***Responding to change*** *over following a plan.*

That is, while there is value in the items on the right, we value the items on the left more. (Agile Alliance 2001)

Lean processes are often associated with agile methods, although they are more scalable and applicable to high-assurance systems. Below, some specific agile methods are presented, and the evolution and content of lean methods is discussed. Please see "Primary References", "Additional References", and the Lean Engineering article for more detail on specific agile and lean processes.

Scrum

Figure 10 shows an example of Scrum as an agile process flow. As with most other agile methods, Scrum uses the evolutionary sequential process shown in Table 1 (above) and described in Fixed-Requirements and Evolutionary Development Processes section in which systems capabilities are developed in short periods, usually around 30 days. The project then re-prioritizes its backlog of desired features and determines how many features the team (usually 10 people or less) can develop in the next 30 days.

Figure 10 also shows that once the features to be developed for the current Scrum have been expanded (usually in the form of informal stories) and allocated to the team members, the team establishes a daily rhythm of starting with a short meeting at which each team member presents a roughly one-minute summary describing progress since the last Scrum meeting, potential obstacles, and plans for the upcoming day.

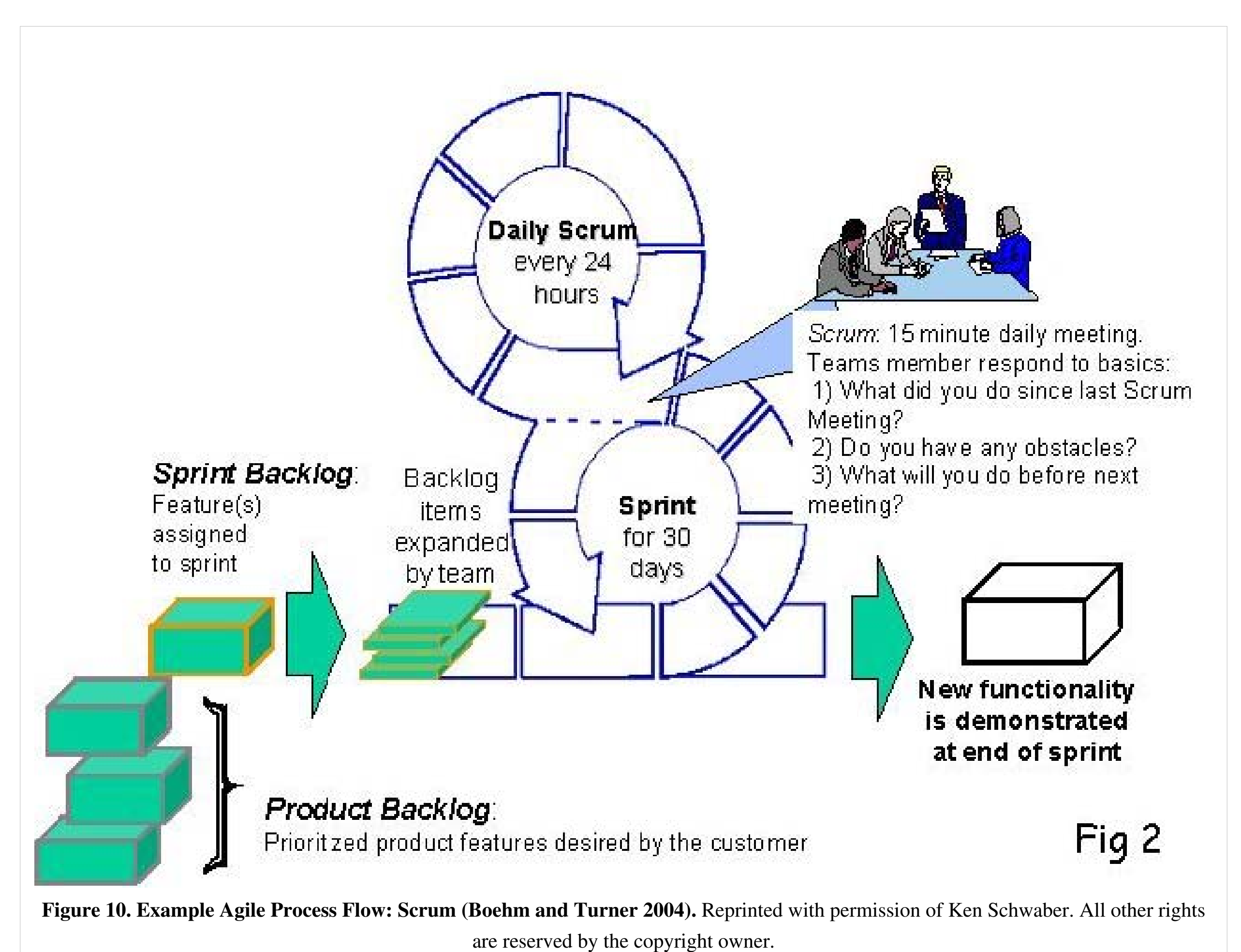


Figure 10. Example Agile Process Flow: Scrum (Boehm and Turner 2004). Reprinted with permission of Ken Schwaber. All other rights are reserved by the copyright owner.

Architected Agile Methods

Over the last decade, several organizations have been able to scale up agile methods by using two layers of ten-person Scrum teams. This involves, among other things, having each Scrum team's daily meeting followed up by a daily meeting of the Scrum team leaders discussing up-front investments in evolving system architecture (Boehm et al. 2010). Figure 11 shows an example of the Architected Agile approach.

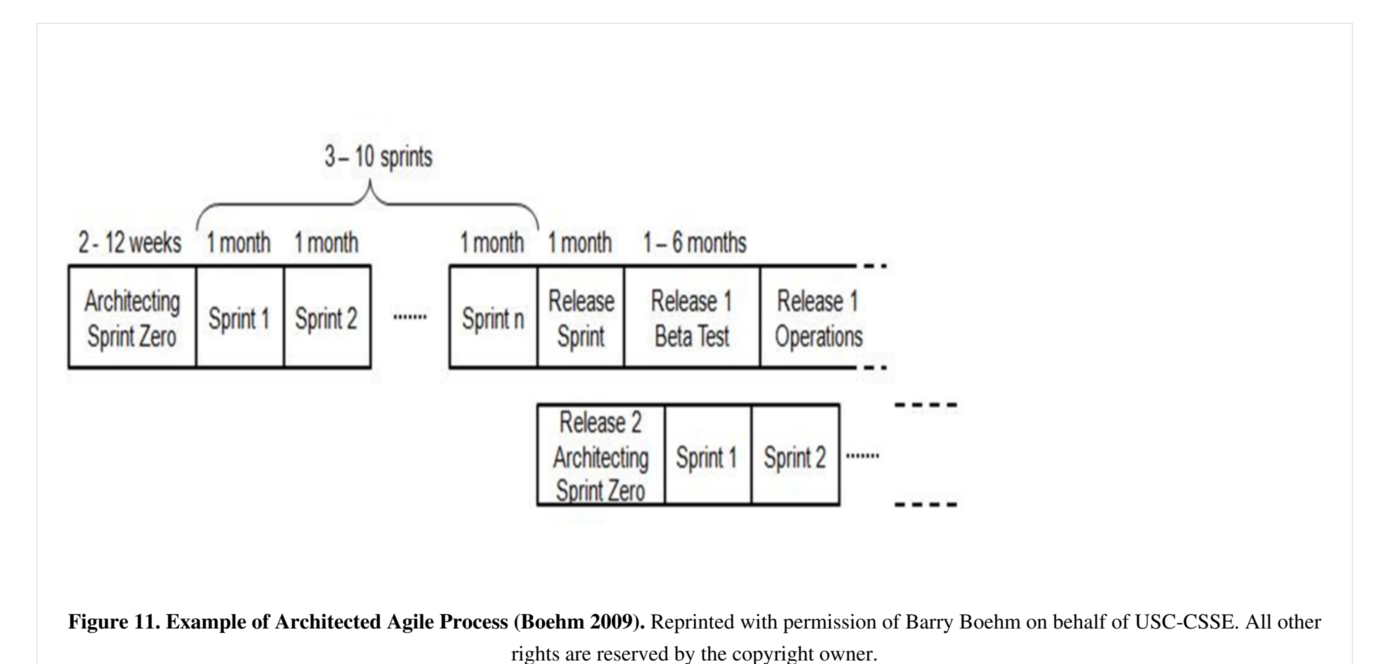


Figure 11. Example of Architected Agile Process (Boehm 2009). Reprinted with permission of Barry Boehm on behalf of USC-CSSE.

Agile Practices and Principles

As seen with the Scrum and architected agile methods, "generally-shared" principles are not necessarily "uniformly followed". However, there are some general practices and principles shared by most agile methods:

- The project team understands, respects, works, and behaves within a defined SE process;
- The project is executed as fast as possible with minimum down time or staff diversion during the project and the critical path is managed;
- All key players are physically or electronically collocated, and "notebooks" are considered team property available to all.
- Baseline management and change control are achieved by formal, oral agreements based on “make a promise—keep a promise” discipline. Participants hold each other accountable.
- Opportunity exploration and risk reduction are accomplished by expert consultation and rapid model verification coupled with close customer collaboration; software development is done in a rapid development environment while hardware is developed in a multi-disciplined model shop; and
- A culture of constructive confrontation pervades the project organization. The team takes ownership for success; it is never “someone else's responsibility.”

Agile development principles (adapted for SE) are as follows (adapted from *Principles behind the Agile Manifesto* (Beedle et al. 2009)):

1. First, satisfy the customer through early and continuous delivery of valuable software (and other system elements).
2. Welcome changing requirements, even late in development; agile processes harness change for the customer's competitive advantage.
3. Deliver working software (and other system elements) frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business personnel and developers must work together daily throughout the project.
5. Build projects around motivated individuals; give them the environment, support their needs, and trust them to get the job done.
6. The most efficient and effective method of conveying information is face-to-face conversation.
7. Working software (and other system elements) is the primary measure of progress.
8. Agile processes promote sustainable development; the sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.

A team should reflect on how to become more effective at regular intervals and then tune and adjust its behavior accordingly. This self-reflection is a critical aspect for projects that implement agile processes.

Lean Systems Engineering and Development

Origins

As the manufacturing of consumer products such as automobiles became more diversified, traditional pre-planned mass-production approaches had increasing problems with quality and adaptability. Lean manufacturing systems such as the Toyota Production System (TPS) (Ohno 1988) were much better suited to accommodate diversity, to improve quality, and to support just-in-time manufacturing that could rapidly adapt to changing demand patterns without having to carry large, expensive inventories.

Much of this transformation was stimulated by the work of W. Edwards Deming, whose Total Quality Management (TQM) approach shifted responsibility for quality and productivity from planners and inspectors to the production workers who were closer to the real processes (Deming 1982). Deming's approach involved everyone in the manufacturing organization in seeking continuous process improvement, or "Kaizen".

Some of the TQM techniques, such as statistical process control and repeatability, were more suited to repetitive manufacturing processes than to knowledge work such as systems engineering (SE) and software engineering (SwE). Others, such as early error elimination, waste elimination, workflow stabilization, and Kaizen, were equally applicable to knowledge work. Led by Watts Humphrey, TQM became the focus for the Software Capability Maturity Model (Humphrey 1987; Paulk et al. 1994) and the CMM-Integrated or CMMI, which extended its scope to include systems engineering (Chrissis et al. 2003). One significant change was the redefinition of Maturity Level 2 from "Repeatable" to "Managed".

The Massachusetts Institute of Technology (MIT) conducted studies of the TPS, which produced a similar approach that was called the "Lean Production System" (Krafcik 1988; Womack et al. 1990). Subsequent development of "lean thinking" and related work at MIT led to the Air Force-sponsored Lean Aerospace Initiative (now called the Lean Advancement Initiative), which applied lean thinking to SE (Murman 2003, Womack-Jones 2003). Concurrently, lean ideas were used to strengthen the scalability and dependability aspects of agile methods for software (Poppendieck 2003; Larman-Vodde 2009). The Kanban flow-oriented approach has been successfully applied to software development (Anderson 2010).

Principles

Each of these efforts has developed a similar but different set of Lean principles. For systems engineering, the current best source is *Lean for Systems Engineering*, the product of several years' work by the INCOSE Lean SE working group (Oppenheim 2011). It is organized into six principles, each of which is elaborated into a set of lean enabler and sub-enabler patterns for satisfying the principle:

1. **Value.** Guide the project by determining the value propositions of the customers and other key stakeholders. Keep them involved and manage changes in their value propositions.
2. **Map the Value Stream (Plan the Program).** This includes thorough requirements specification, the concurrent exploration of trade spaces among the value propositions, COTS evaluation, and technology maturity assessment, resulting in a full project plan and set of requirements.
3. **Flow.** Focus on careful attention to the project's critical path activities to avoid expensive work stoppages, including coordination with external suppliers.
4. **Pull.** Pull the next tasks to be done based on prioritized needs and dependencies. If a need for the task can't be found, reject it as waste.
5. **Perfection.** Apply continuous process improvement to approach perfection. Drive defects out early to get the system Right The First #**Time,** vs. fixing them during inspection and test. Find and fix root causes rather than symptoms.
6. **Respect for People.** Flow down responsibility, authority, and accountability to all personnel. Nurture a learning environment. Treat people as the organization's most valued assets. (Oppenheim 2011)

These lean SE principles are highly similar to the four underlying incremental commitment spiral model principles.

- **Principle 1: Stakeholder value-based system definition and evolution**, addresses the lean SE principles of value, value stream mapping, and respect for people (developers are success-critical stakeholders in the ICSM).
- **Principle 2: Incremental commitment and accountability**, partly addresses the pull principle, and also addresses respect for people (who are accountable for their commitments).
- **Principle 3: Concurrent system and software definition and development**, partly addresses both value stream mapping and flow.
- **Principle 4: Evidence and risk-based decision making**, uses evidence of achievability as its measure of success. Overall, the ICSM principles are somewhat light on continuous process improvement, and the lean SE principles are somewhat insensitive to requirements emergence in advocating a full pre-specified project plan and set of requirements.

See Lean Engineering for more information.

References

Works Cited

Agile Alliance. 2001. "Manifesto for Agile Software Development." http://agilemanifesto.org/.

Anderson, D. 2010. *Kanban*, Sequim, WA: Blue Hole Press.

Boehm, B. 1996. "Anchoring the Software Process." IEEE *Software* 13(4): 73-82.

Boehm, B. and J. Lane. 2007. "Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering." *CrossTalk.* 20(10) (October 2007): 4-9.

Boehm, B., J. Lane, S. Koolmanjwong, and R. Turner. 2010. "Architected Agile Solutions for Software-Reliant Systems," in Dingsoyr, T., T. Dyba., and N. Moe (eds.), *Agile Software Development: Current Research and Future Directions.* New York, NY, USA: Springer.

Boehm, B. and R. Turner. 2004. *Balancing Agility and Discipline.* New York, NY, USA: Addison-Wesley.

Castellano, D.R. 2004. “Top Five Quality Software Projects.” *CrossTalk.* 17(7) (July 2004): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2004/200407/200407-0-Issue.pdf

Chrissis, M., M. Konrad, and S. Shrum. 2003. *CMMI: Guidelines for Process Integration and Product Improvement.* New York, NY, USA, Addison Wesley.

Deming, W.E. 1982. *Out of the Crisis.* Cambridge, MA, USA: MIT.

Fairley, R. 2009. *Managing and Leading Software Projects.* New York, NY, USA: John Wiley & Sons.

Forsberg, K. 1995. "If I Could Do That, Then I Could…’ System Engineering in a Research and Development Environment." Proceedings of the Fifth International Council on Systems Engineering (INCOSE) International Symposium. 22-26 July 1995. St Louis, MO, USA.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Humphrey, W., 1987. “Characterizing the Software Process: A Maturity Framework.” Pittsburgh, PA, USA: CMU Software Engineering Institute. CMU/SEI-87-TR-11.

Jarzombek, J. 2003. “Top Five Quality Software Projects.” *CrossTalk.* 16(7) (July 2003): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2003/200307/200307-0-Issue.pdf.

Krafcik, J. 1988. "Triumph of the lean production system". *Sloan Management Review.* 30(1): 41–52.

Kruchten, P. 1999. *The Rational Unified Process*. New York, NY, USA: Addison Wesley.

Larman , C. and B. Vodde. 2009. *Scaling Lean and Agile Development.* New York, NY, USA: Addison Wesley.

Maranzano, J.F., S.A. Rozsypal, G.H. Zimmerman, G.W. Warnken, P.E. Wirth, D.M. Weiss. 2005. “Architecture Reviews: Practice and Experience.” IEEE *Software.* 22(2): 34-43.

Murman, E. 2003. *Lean Systems Engineering I, II, Lecture Notes*, MIT Course 16.885J, Fall. Cambridge, MA, USA: MIT.

Oppenheim, B. 2011. *Lean for Systems Engineering.* Hoboken, NJ: Wiley.

Paulk, M., C. Weber, B. Curtis, and M. Chrissis. 1994. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Reading, MA, USA: Addison Wesley.

Pew, R. and A. Mavor (eds.). 2007. *Human-System Integration in The System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Poppendieck, M. and T. Poppendieck. 2003. *Lean Software Development: An Agile Toolkit for Software Development Managers.* New York, NY, USA: Addison Wesley.

Spruill, N. 2002. “Top Five Quality Software Projects.” *CrossTalk.* 15(1) (January 2002): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2002/200201/200201-0-Issue.pdf.

Stauder, T. “Top Five Department of Defense Program Awards.” *CrossTalk.* 18(9) (September 2005): 4-13. Available at http://www.crosstalkonline.org/storage/issue-archives/2005/200509/200509-0-Issue.pdf.

Womack, J., D. Jones, and D Roos. 1990. *The Machine That Changed the World: The Story of Lean Production.* New York, NY, USA: Rawson Associates.

Womack, J. and D. Jones. 2003. *Lean Thinking*. New York, NY, USA: The Free Press.

Primary References

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto". in *The Agile Manifesto* [database online]. Accessed 2010. Available at: www.agilemanifesto.org/principles.html

Boehm, B. and R. Turner. 2004. *Balancing Agility and Discipline.* New York, NY, USA: Addison-Wesley.

Fairley, R. 2009. *Managing and Leading Software Projects.* New York, NY, USA: J. Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* Kings College, UK: College Publications.

Pew, R., and A. Mavor (eds.). 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Royce, W.E. 1998. *Software Project Management: A Unified Framework*. New York, NY, USA: Addison Wesley.

Additional References

Anderson, D. 2010. *Kanban*. Sequim, WA, USA: Blue Hole Press.

Baldwin, C. and K. Clark. 2000. *Design Rules: The Power of Modularity.* Cambridge, MA, USA: MIT Press.

Beck, K. 1999. *Extreme Programming Explained.* New York, NY, USA: Addison Wesley.

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto" in The Agile Manifesto [database online]. Accessed 2010. Available at: www.agilemanifesto.org/principles.html

Biffl, S., A. Aurum, B. Boehm, H. Erdogmus, and P. Gruenbacher (eds.). 2005. *Value-Based Software Engineering*. New York, NY, USA: Springer.

Boehm, B. 1988. “A Spiral Model of Software Development.” IEEE *Computer.* 21(5): 61-72.

Boehm, B. 2006. “Some Future Trends and Implications for Systems and Software Engineering Processes.” *Systems Engineering.* 9(1): 1-19.

Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy. 1998. “Using the WinWin Spiral Model: A Case Study.” IEEE *Computer.* 31(7): 33-44.

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner. 2013 (in press). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model.* New York, NY, USA: Addison Wesley.

Castellano, D.R. 2004. “Top Five Quality Software Projects.” *CrossTalk.* 17(7) (July 2004): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2004/200407/200407-0-Issue.pdf

Checkland, P. 1981. *Systems Thinking, Systems Practice*. New York, NY, USA: Wiley.

Crosson, S. and B. Boehm. 2009. “Adjusting Software Life Cycle Anchorpoints: Lessons Learned in a System of Systems Context.” Proceedings of the Systems and Software Technology Conference, 20-23 April 2009, Salt Lake City, UT, USA.

Dingsoyr, T., T. Dyba. and N. Moe (eds.). 2010. "Agile Software Development: Current Research and Future Directions.” Chapter in B. Boehm, J. Lane, S. Koolmanjwong, and R. Turner, *Architected Agile Solutions for Software-Reliant Systems.* New York, NY, USA: Springer.

Dorner, D. 1996. *The Logic of Failure*. New York, NY, USA: Basic Books.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Forsberg, K. 1995. "'If I Could Do That, Then I Could…' System Engineering in a Research and Development Environment." Proceedings of the Fifth Annual International Council on Systems Engineering (INCOSE) International Symposium. 22-26 July 1995. St. Louis, MO, USA.

Forsberg, K. 2010. "Projects Don't Begin With Requirements." Proceedings of the IEEE Systems Conference. 5-8 April 2010. San Diego, CA, USA.

Gilb, T. 2005. *Competitive Engineering*. Maryland Heights, MO, USA: Elsevier Butterworth Heinemann.

Goldratt, E. 1984. *The Goal*. Great Barrington, MA, USA: North River Press.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* New York, NY, USA: Wiley.

Holland, J. 1998. *Emergence*. New York, NY, USA: Perseus Books.

ISO/IEC. 2010. *Systems and Software Engineering, Part 1: Guide for Life Cycle Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 24748-1:2010.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes.* Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

ISO/IEC. 2003. *Systems Engineering — A Guide for The Application of ISO/IEC 15288 System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 19760:2003 (E).

Jarzombek, J. 2003. "Top Five Quality Software Projects." *CrossTalk.* 16(7) (July 2003): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2003/200307/200307-0-Issue.pdf.

Kruchten, P. 1999. *The Rational Unified Process.* New York, NY, USA: Addison Wesley.

Landis, T. R. 2010. *Lockheed Blackbird Family (A-12, YF-12, D-21/M-21 & SR-71).* North Branch, MN, USA: Specialty Press.

Madachy, R. 2008. *Software Process Dynamics*. New York, NY, USA: Wiley.

Maranzano, J., et al. 2005. "Architecture Reviews: Practice and Experience." IEEE *Software.* 22(2): 34-43.

National Research Council of the National Academies (USA). 2008. *Pre-Milestone A and Early-Phase Systems Engineering*. Washington, DC, USA: The National Academies Press.

Osterweil, L. 1987. "Software Processes are Software Too." Proceedings of the SEFM 2011: 9th International Conference on Software Engineering. Monterey, CA, USA.

Poppendeick, M. and T. Poppendeick. 2003. *Lean Software Development: an Agile Toolkit.* New York, NY, USA: Addison Wesley.

Rechtin, E. 1991. *System Architecting: Creating and Building Complex Systems.* Upper Saddle River, NY, USA: Prentice-Hall.

Rechtin, E., and M. Maier. 1997. *The Art of System Architecting*. Boca Raton, FL, USA: CRC Press.

Schwaber, K. and M. Beedle. 2002. *Agile Software Development with Scrum*. Upper Saddle River, NY, USA: Prentice Hall.

Spruill, N. 2002. "Top Five Quality Software Projects." *CrossTalk.* 15(1) (January 2002): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2002/200201/200201-0-Issue.pdf.

Stauder, T. 2005. "Top Five Department of Defense Program Awards." *CrossTalk.* 18(9) (September 2005): 4-13. Available at http://www.crosstalkonline.org/storage/issue-archives/2005/200509/200509-0-Issue.pdf.

Warfield, J. 1976. *Societal Systems: Planning, Policy, and Complexity*. New York, NY, USA: Wiley.

Womack, J. and D. Jones. 1996. *Lean Thinking.* New York, NY, USA: Simon and Schuster.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Integration of Process and Product Models

When performing systems engineering activities, it is important to consider the mutual relationship between processes and the desired system. The type of system (see Types of Systems) being produced will affect the needed processes, as indicated in system life cycle process drivers and choices. This may cause the tailoring (glossary) of defined processes as described in application of systems engineering standards.

Process and Product Models

Figure 1 of life cycle models introduced the perspective of viewing stage work products provided by process execution as versions of a system-of-interest (SoI) at various life stages. The fundamental changes that take place during the life cycle of any man-made system include definition, production, and utilization. Building upon these, it is useful to consider the structure of a generic process and product life cycle stage model as portrayed in Figure 1 below.

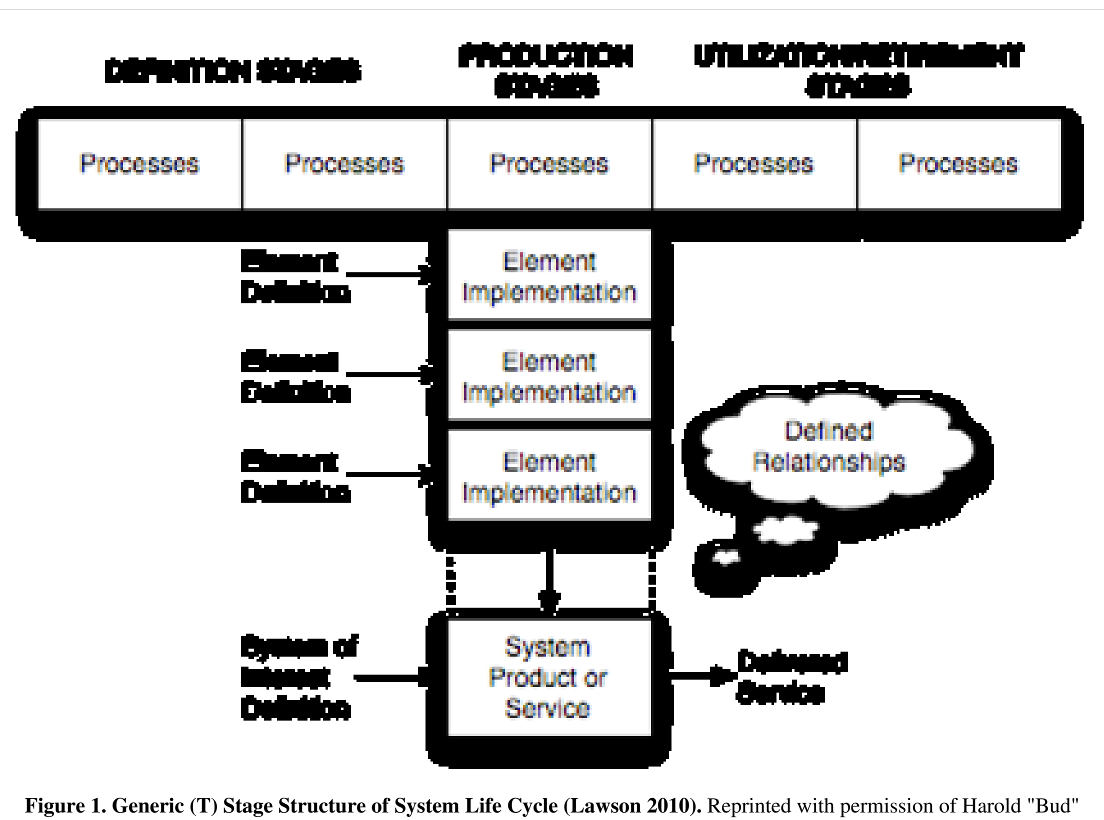


Figure 1. Generic (T) Stage Structure of System Life Cycle (Lawson 2010). Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

The (T) model indicates that a definition stage precedes a production stage where the implementation (acquisition, provisioning, or development) of two or more system elements has been accomplished. The system elements are integrated according to defined relationships into the SoI. Thus, both the process and product aspects are portrayed. The implementation and integration processes are followed in providing the primary stage results—namely, in assembled system product or service instances. However, as noted in life cycle models, the definition of the SoI when provided in a development stage can also be the result of first versions of the system. For example, a prototype (glossary), which may be viewed as a form of production or pre-production stage. Following the production stage is

a utilization stage. Further relevant stages can include support and retirement. Note that this model also displays the important distinction between definition versus implementation and integration.

According to ISO/IEC/IEEE 15288 (2015), this structure is generic for any type of man-made SoI to undergo life cycle management. The production stage thus becomes the focal point of the (T) model at which system elements are implemented and integrated into system product or service instances based upon the definitions. For defined physical systems, this is the point at which product instances are manufactured and assembled (singularly or mass-produced). For non-physical systems, the implementation and integration processes are used in service preparation (establishment) prior to being instantiated to provide a service. For software systems, this is the point at which builds that combine software elements into versions, releases, or some other form of managed software product are produced.

Using recursive decomposition, the implementation of each system element can involve the invocation of the standard again at the next lowest level, thus treating the system element as a SoI in its own right. A new life cycle structure is then utilized for the lower level SoIs.

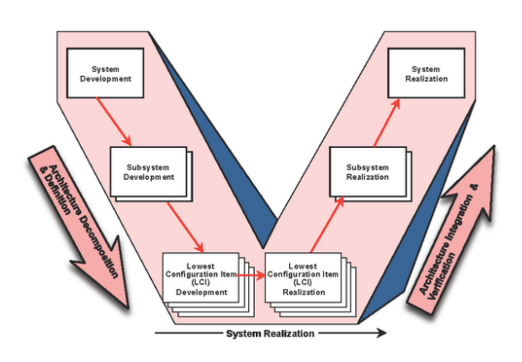
This is illustrated in the Dual Vee model (Figures 2a and 2b). The Dual Vee model is a three-dimensional system development model that integrates product and process in the creation of the system and component architectures. It emphasizes

- concurrent opportunity and risk management;
- user in-process validation (glossary);
- integration (glossary), verification (glossary), and validation (glossary) planning; and
- verification (glossary) problem resolution.

When decomposition terminates according to the practical need and risk-benefit analysis, system elements are then implemented (acquired, provisioned, or developed) according to the type of element involved.

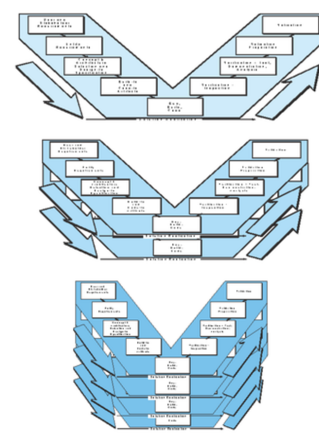
Dual Vee Model

Architecture Vee
for architecture management



Depicts architecture baseline evolution. Vertical dimension is architecture decomposition. Horizontal dimension is system realization. Third dimension normal to the image is quantity of entities and their interfaces.

Entity Vee
for entity management



An entity is any item of the architecture. The Vee depicts entity baseline elaboration. Vertical dimension is entity detail. Horizontal dimension is entity realization.

Figure 2a. The Dual Vee Model (2a) (Forsberg, Mooz, Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc.

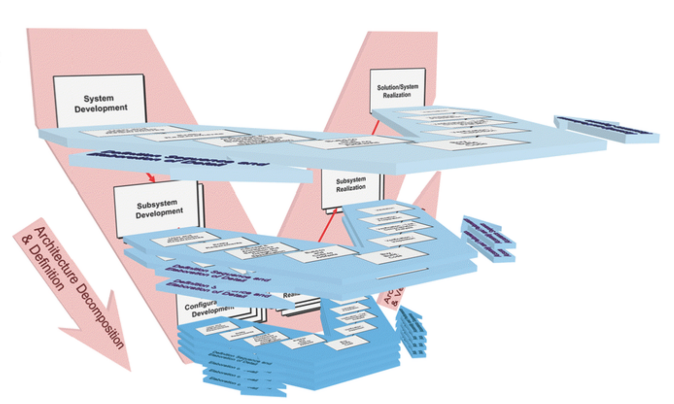


Figure 2b. The Dual Vee Model (2b) (Forsberg, Mooz, Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

A practical aspect that can very well affect the process and product aspect is the decision to use off-the-shelf elements in commercial-off-the-shelf (COTS) form. In this case, further decomposition of the element is not necessary. The use of COTS elements (and their internally created neighbor or non-development item (NDI)) has become widespread, and they have proven their value. However, developers must make sure that the COTS product is appropriate for their environment.

A known flaw which occurs infrequently in normal use of the product in its intended environment may be benign and easily dealt with. In a new situation, it could have dramatic adverse consequences, such as those that occurred on the USS Yorktown Cruiser in 1998 (Wired News Contributors 1998). The customer mandated that Windows NT be used as the primary operating system for the ship. A *divide by zero* fault caused the operating system to fail, and the ship was dead in the water. It had to be towed back to port on three occasions.

Spiral models concurrently engineer not only process and product models, but also property and success models. Figure 3 shows how these models provide checks and balances, both at milestone reviews and as individual model choices are made. Methods and tools supporting this concurrent engineering are provided in “When Models Collide: Lessons from Software System Analysis” (Boehm and Port 1999), “Avoiding the Software Model-Clash Spiderweb” (Boehm, Port, and Al-Said 2000), and “Detecting Model Clashes During Software Systems Development” (Al-Said 2003).

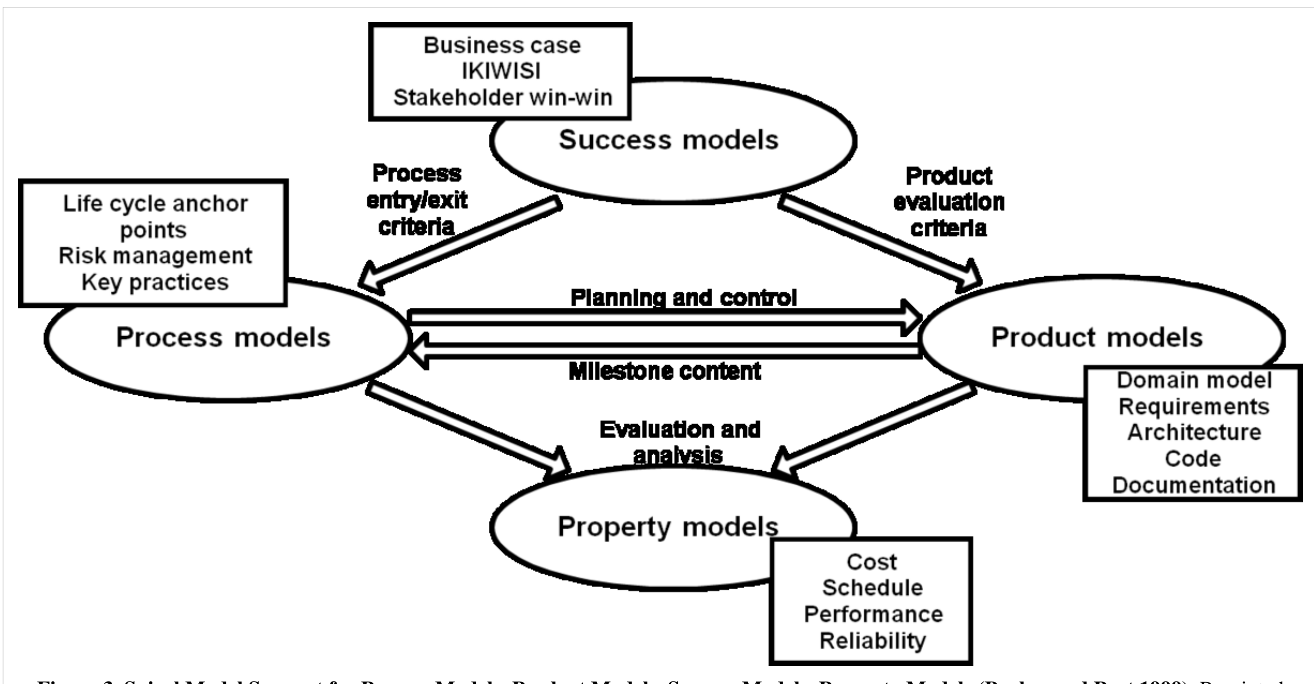


Figure 3. Spiral Model Support for Process Models, Product Models, Success Models, Property Models (Boehm and Port 1999). Reprinted with permission of © Copyright IEEE – All rights reserved. All other rights are reserved by the copyright owner.

For software systems, entry into the production stages is the point at which builds that combine software elements (code modules) into versions, releases, or some other form of managed software product are created. Thus, the major difference between systems in general and software systems is the slight variant of the generic model as presented in Figure 4.

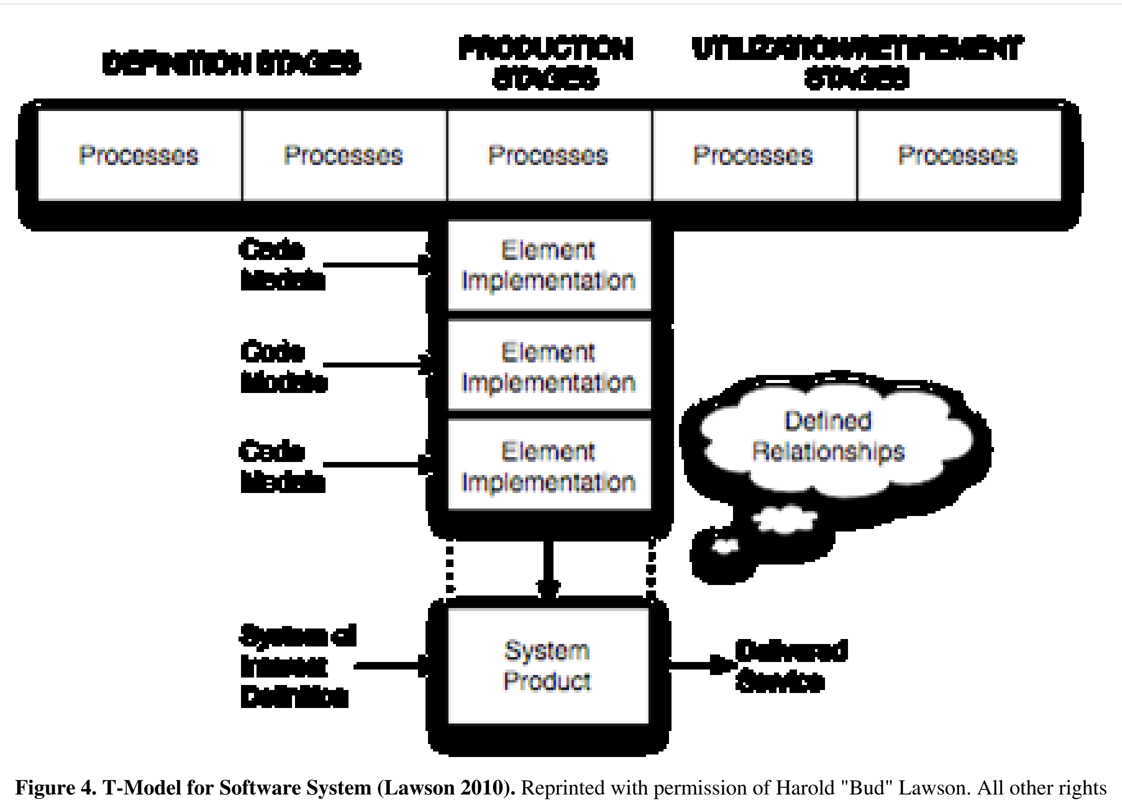


Figure 4. T-Model for Software System (Lawson 2010). Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

Stage Execution Order

A sequential execution of life cycle stages is the most straightforward. As presented in System Life Cycle Process Models: Vee and System Life Cycle Process Models: Iterative, variants of the Vee model and the spiral model provide non-sequential models when practical considerations require a non-linear execution of life cycle stages. Building upon these two models, it is important to note that various types of complex systems require that the stages of the life cycle model be revisited as insight (knowledge) is gained, as well as when stakeholder requirements change. The iterations may involve necessary changes in the processes and in the product or service system. Thus, within the context of the (T) stage model, various orderings of stage execution - reflecting forms of non-sequential stage ordering - can be conveniently described, as portrayed in Figure 5.

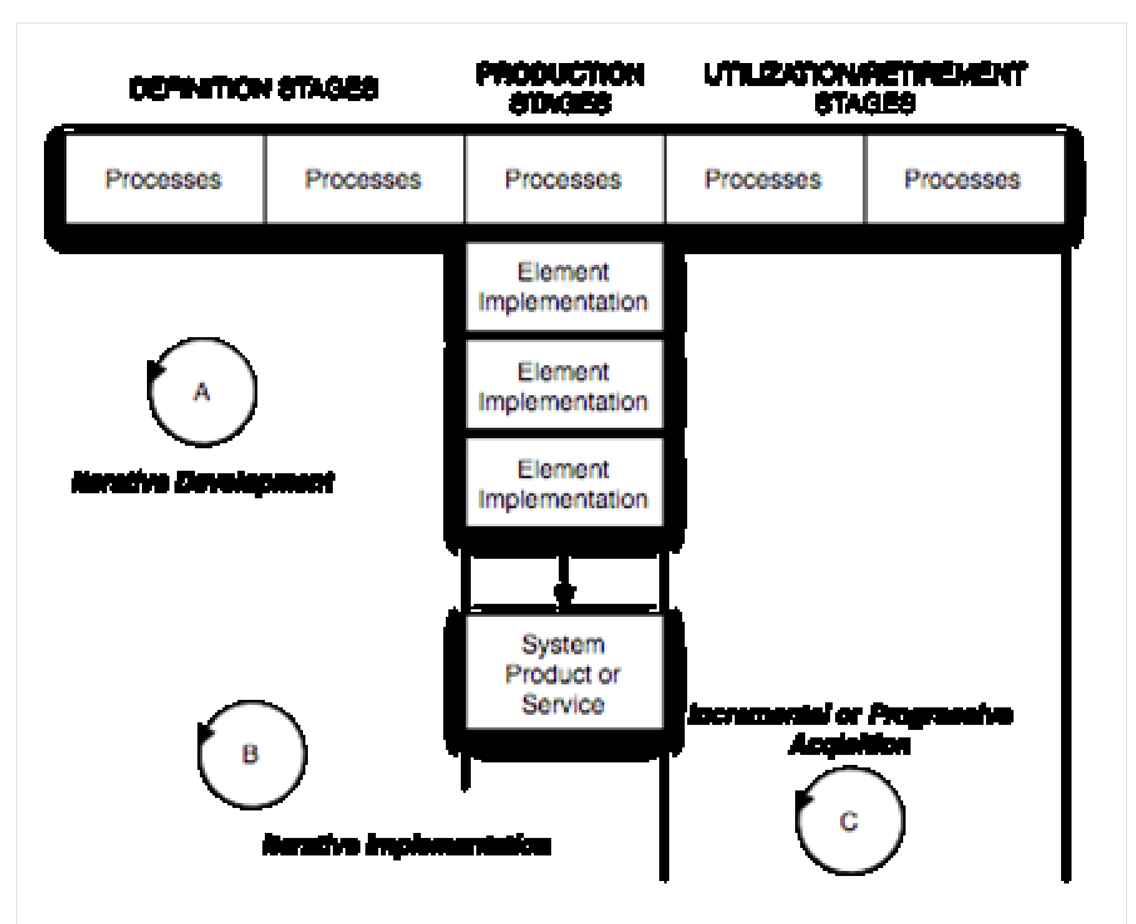


Figure 5. Iteration Through Life Cycle Stages (Lawson 2010). Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

Each pattern of stage execution involves iteration of the previous stages, perhaps with altered requirements for the processes or the system. The heavy lines in Figure 5 denote the demarcation of the revisited end points. Three are iterative forms, for which several variants can be extracted:

1. **Iterative development** is quite frequently deployed in order to assess stakeholder requirements, analyze the requirements, and develop a viable architectural design. Thus, it is typical that the concept stage may be revisited during the development stage. For systems where products are based upon physical structures (electronics, mechanics, chemicals, and so on), iteration after production has begun can involve significant costs and schedule delays. It is, therefore, important to get it *"right"* before going to production. The early stages are thus used to build confidence (verify and validate) that the solution works properly and will meet the needs of the stakeholders. Naturally, such an approach could be used for software and human activity systems as well; however, due to their soft nature, it can be useful to go further by experimenting and evaluating various configurations of the system.

2. **Iterative development and implementation** involves producing (defining, implementing, and integrating) various versions of the system, evaluating how well they meet stakeholder requirements, perhaps in the context of changing requirements, and then revisiting the concept and/or development stages. Such iterations are typical within software system development, where the cost of production is not as significant as for defined physical systems. A variant of this approach is the spiral model, where successive iterations fill in more detail (Boehm and May 1998). The use of this approach requires careful attention to issues related to baseline and configuration management. In this approach, significant verification (testing) should be performed on software systems in order to build confidence that the system delivered will meet stakeholder requirements.
3. **Incremental or progressive acquisition** involves releasing systems in the form of products and/or services to the consumers. This approach is appropriate when structural and capability (functions) changes are anticipated in a controlled manner after deployment. The use of this approach can be due to not knowing all of the requirements at the beginning, which leads to progressive acquisition/deployment, or due to a decision to handle the complexity of the system and its utilization in increments—namely, incremental acquisition. These approaches are vital for complex systems in which software is a significant system element. Each increment involves revisiting the definition and production stages. The utilization of these approaches must be based upon well-defined, agreed relationships between the supplying and acquiring enterprises. In fact, the iteration associated with each resulting product and/or service instance may well be viewed as a joint project, with actor roles being provided by both enterprises.

In all of the approaches it is wise to use modeling and simulation techniques and related tools to assist in understanding the effect of changes made in the complex systems being life cycle managed. These techniques are typically deployed in the earlier stages; however, they can be used in gaining insight into the potential problems and opportunities associated with the latter stages of utilization and maintenance (for example, in understanding the required logistics and help-desk aspects).

Allocating and Meeting Requirements - Integration of Process and Product Models

Regardless of the order in which life cycle stages are executed, stakeholder requirements for the system, including changed requirements in each iteration, must be allocated into appropriate activities of the processes used in projects for various stages as well as to the properties of the elements of the product system or service system and their defined relationships. This distribution was illustrated in the fourth variant of Lawson’s T-model as presented in System Life Cycle Process Models: Iterative and System Life Cycle Process Models: Vee.

Ideally, the project management team should implement proven processes that will integrate the technical process models with the project management product models to manage any of the processes discussed earlier, including incremental and evolutionary development. The processes shown are the project management flow, starting with the beginning of the development phase (Forsberg, Mooz, and Cotterman 2005, 201).

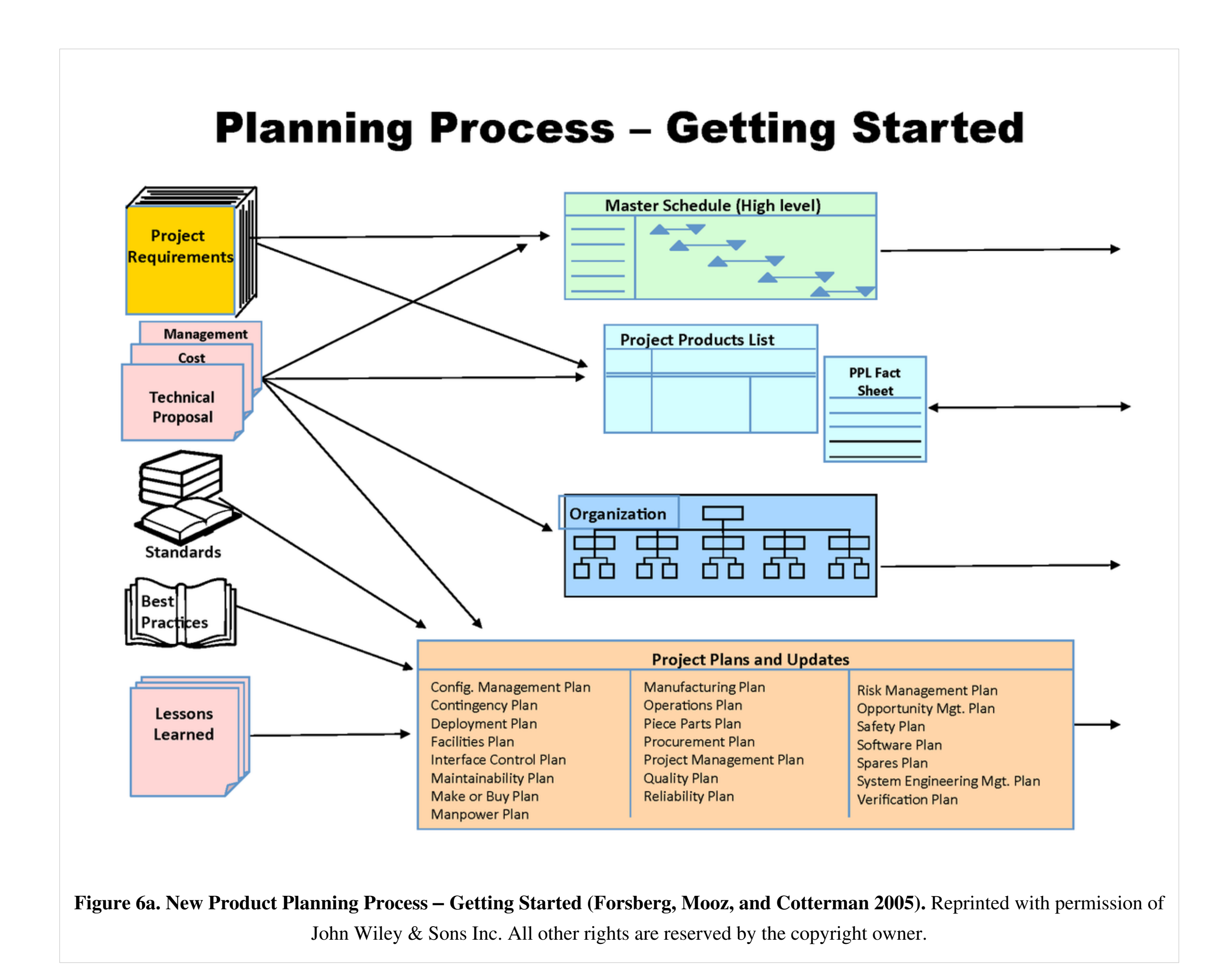


Figure 6a. New Product Planning Process – Getting Started (Forsberg, Mooz, and Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc.

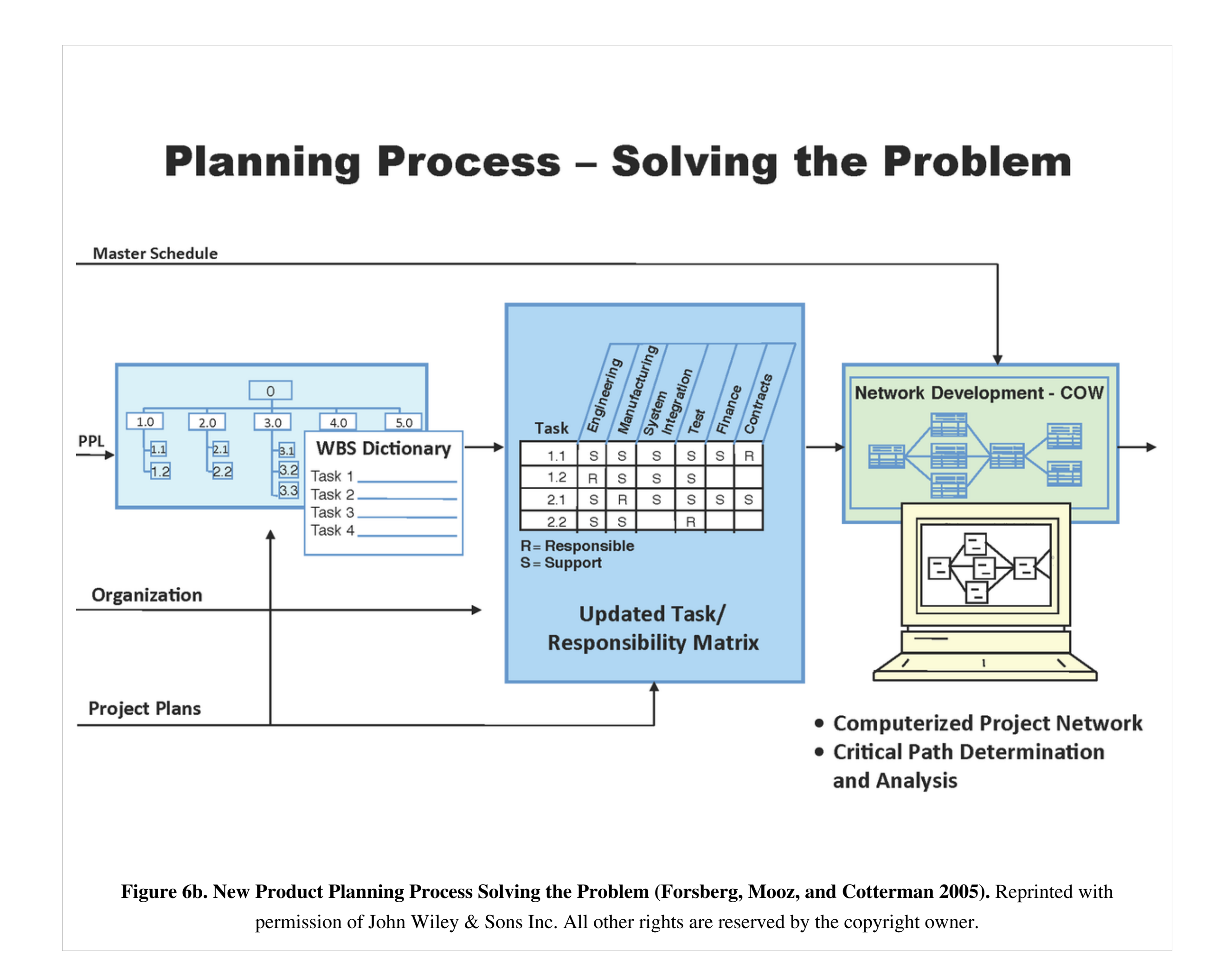


Figure 6b. New Product Planning Process Solving the Problem (Forsberg, Mooz, and Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

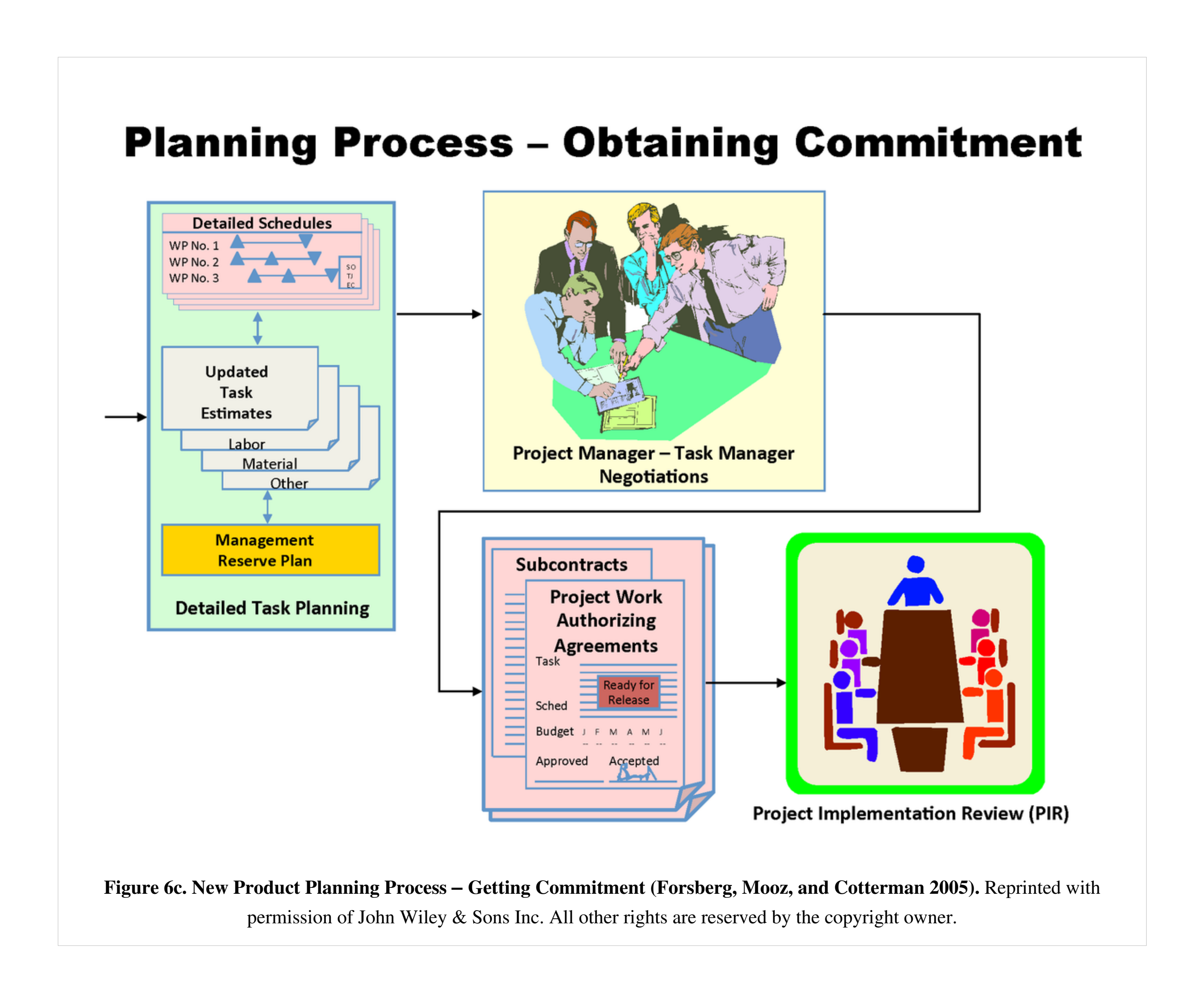


Figure 6c. New Product Planning Process – Getting Commitment (Forsberg, Mooz, and Cotterman 2005). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

References

Works Cited

Boehm, B. and W. May. 1988. "A Spiral Model of Software Development and Enhancement." IEEE *Computer* 21(5): 61-72.

Boehm, B. and D. Port. 1999. "When Models Collide: Lessons From Software System Analysis." *IT Professional* 1(1): 49-56.

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner (forthcoming). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model.* New York, NY, USA: Addison Wesley.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* 3rd ed. New York, NY, USA: J. Wiley & Sons.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering-- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions.ISO/IEC/IEEE 15288:2015

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Wired News Contributors. 2011. “Sunk by Windows NT,” *Wired News*, last modified July 24, 1998. Accessed on September 11, 2011. Available at http://www.wired.com/science/discoveries/news/1998/07/13987.

Primary References

Boehm, B. and W. May. 1988. “A Spiral Model of Software Development and Enhancement.” IEEE *Computer.* 21(5): 61-72.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Additional References

Al-Said, M. 2003. "Detecting Model Clashes During Software Systems Development." PhD Diss. Department of Computer Science, University of Southern California, December 2003.

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner. (forthcoming). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model.* New York, NY, USA: Addison Wesley.

Boehm, B. and D. Port. 1999. "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them." *ACM Software Engineering Notes.* (January, 1999): p. 36-48.

Boehm, B. and D. Port. 1999. "When Models Collide: Lessons From Software System Analysis." *IT Professional.* 1(1): 49-56.

Boehm, B., D. Port, and M. Al-Said. 2000. "Avoiding the Software Model-Clash Spiderweb." IEEE *Computer.* 33(11): 120-122.

Lawson, H. and M. Persson. 2010. “Portraying Aspects of System Life Cycle Models.” Proceedings of the European Systems Engineering Conference (EuSEC). 23-26 May 2010. Stockholm, Sweden.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Lean Engineering

Lean Systems Engineering (LSE) is the application of lean thinking (Womack 2003) to systems engineering (SE) and related aspects of enterprise and project management. LSE is an approach that is applicable throughout the system life cycle. The goal of LSE is to deliver the best life-cycle value for technically complex systems with minimal waste. Lean engineering is relevant to all of the traditional SE technical processes (see concept definition, system definition, system realization, system deployment and use, etc.). Lean engineering also interacts with and utilizes many of the specialty engineering disciplines discussed in Part 6.

Lean Systems Engineering

SE is an established, sound practice, but not always delivered effectively. Most programs are burdened with some form of waste such as: poor coordination, unstable requirements, quality problems, delays, rework, or management frustration. Recent U.S. Government Accountability Office (GAO), National Aeronautics and Space Association (NASA), and Massachusetts Institute of Technology (MIT) studies of government programs document major budget and schedule overruns and a significant amount of waste in government programs - some reaching seventy percent of charged time. This waste represents a productivity reserve in programs and major opportunities to improve program efficiency.

LSE is the application of lean thinking to systems engineering and related aspects of enterprise and project management. SE is focused on the discipline that enables development of complex technical systems. Lean thinking is a holistic paradigm that focuses on delivering maximum value to the customer and minimizing wasteful practices. It has been successfully applied in manufacturing, aircraft depots, administration, supply chain management, healthcare, and product development, which includes engineering. LSE is the area of synergy between lean thinking and SE, which aims to deliver the best life-cycle value for technically complex systems with minimal waste. LSE does not mean less SE. It means more and better SE with higher responsibility, authority, and accountability (RAA), leading to better, waste-free workflow with increased mission assurance. Under the LSE philosophy, mission assurance is non-negotiable and any task which is legitimately required for success must be included; however, it should be well-planned and executed with minimal waste.

Lean Principles

Oppenheim (2011) describes the six lean principles for product development (PD) as follows:

1. **Capture the *value* defined by the customer.** One cannot over-emphasize the importance of capturing task or program value (requirements, CONOPS, etc.) with precision, clarity, and completeness before resource expenditures ramp up to avoid unnecessary rework.
2. ***Map the value stream (plan the program)* and eliminate waste.** Map all end-to-end linked tasks, control/decision nodes, and the interconnecting information flows necessary to realize customer value. During the mapping process, eliminate all non-value added activities, and enable the remaining activities to flow (without rework, backflow or stopping). The term *information flow* refers to the packets of information (knowledge) created by different tasks and flowing to other tasks for subsequent value adding, such as: design, analysis, test, review, decision, or integration. Each task adds value if it increases the level of useful information and reduces risk in the context of delivering customer value.
3. ***Flow* the work through planned and streamlined value - adding steps and processes, without stopping or idle time, unplanned rework, or backflow.** To optimize flow, one should plan for maximum concurrency of tasks, up to near capacity of an enterprise. Legitimate engineering iterations are frequently needed in PD, but they tend to be time consuming and expensive if they extend across disciplines. Lean PD encourages efficient methodology of *fail early - fail often* through rapid architecting and discovery techniques during early design

phases. Lean flow also makes every effort to use techniques that prevent lengthy iterations, such as design frontloading, trade space explorations, set designs, modular designs, legacy knowledge, and large margins. Where detailed cross-functional iterations are indeed necessary, lean flow optimizes iteration loops for overall value.

4. **Let customers *pull* value.** In PD, the pull principle has two important meanings: (1) the inclusion of any task in a program must be justified by a specific need from an internal or external customer and coordinated with them, and (2) the task should be completed when the customer needs the output. Excessively early completion leads to “shelf life obsolescence” including possible loss of human memory or changed requirements and late completion leads to schedule slips. This is the reason that every task owner or engineer needs to be in close communication with their internal customers to fully understand their needs and expectations and to coordinate their work.
5. **Pursue *perfection* of all processes.** Global competition requires continuous improvements of processes and products. Yet, no organization can afford to spend resources improving everything all the time. Systems engineers must make a distinction between processes and process outputs. Perfecting and refining the *work output* in a given task must be bounded by the overall value proposition (system or mission success, program budget and schedule) which define when an output is "good enough". In contrast, engineering and other *processes* must be continuously improved for competitive reasons.
6. ***Respect* for people.** A lean enterprise is an organization that recognizes that its people are the most important resource. In a lean enterprise, people are not afraid to identify problems and imperfections honestly and openly in real time, brainstorm about root causes and corrective actions without fear, or plan effective solutions together by consensus to prevent a problem from occurring again.

Lean Enablers for Systems

In 2009, the International Council on Systems Engineering's (INCOSE's) Lean SE Working Group (LSE WG) released an online product entitled *Lean Enablers for Systems Engineering (LEfSE)*. It is a collection of practices and recommendations formulated as “dos” and “don’ts” of SE, based on lean thinking. The practices cover a large spectrum of SE and other relevant enterprise management practices, with a general focus on improving the program value and stakeholder satisfaction and reduce waste, delays, cost overruns, and frustrations. LEfSE are grouped under the six lean principles outlined above. The LEfSE are not intended to become a mandatory practice but should be used as a checklist of good practices. LEfSE do not replace the traditional SE; instead, they amend it with lean thinking.

LEfSE were developed by fourteen experienced INCOSE practitioners, some recognized leaders in lean and SE from industry, academia, and governments (such as the U.S., United Kingdom, and Israel), with cooperation from the 160-member international LSE WG. They collected best practices from the many companies, added collective tacit knowledge, wisdom, and experience of the LSE WG members, and inserted best practices from lean research and literature. The product has been evaluated by surveys and comparisons with the recent programmatic recommendations by GAO and NASA.

Oppenheim (2011) includes a comprehensive explanation of the enablers, as well as the history of LSE, the development process of LEfSE, industrial examples, and other material. Oppeneheim, Murman, and Secor (2011) provide a scholarly article about LEfSE. A short summary was also published by Oppenheim in 2009.

References

Works Cited

Lean Systems Engineering Working Group. 2009. "Lean Enablers for Systems Engineering." Accessed 13 January 2016 at http:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/ 07/ Lean-Enablers-for-SE-Version-1_03-.pdf

Lean Systems Engineering Working Group. 2009. "Quick Reference Guide Lean Enablers for Systems Engineering." Accessed 13 January 2016 at http:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/ 07/ LEfSE-Quick-Reference-Guide-8-pages.pdf

Oppenheim, B.W. 2009. "Process Replication: Lean Enablers for Systems Engineering." *CrossTalk, The Journal of Defense Software Engineering*. July/August 2009.

Oppenheim, B.W. 2011. *Lean for Systems Engineering, with Lean Enablers for Systems Engineering*. Hoboken, NJ, USA: Wiley.

Oppenheim, B.W., E.M. Murman, and D. Secor. 2011. "Lean Enablers for Systems Engineering." *Journal of Systems Engineering*. 1(14).

Womack, J.P. 2003. *Lean Thinking*. Columbus, OH, USA: Free Press.

Primary References

Lean Systems Engineering Working Group. 2009. "Lean Enablers for Systems Engineering." Accessed 13 January 2016 athttp:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/ 07/ Lean-Enablers-for-SE-Version-1_03-.pdf

Oppenheim, B., E. Murman, and D. Sekor. 2010. "Lean Enablers for Systems Engineering." Systems Engineering. 14(1). Accessed 13 January 2016. Available at http:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/07/LEfSE-JSE-compressed.pdf

Additional References

Lean Enterprise Institute. 2009. "Principles of Lean." Accessed 1 March 2012 at http://www.lean.org/WhatsLean/ Principles.cfm.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Concept Defnition

Concept Definition

Concept Definition (glossary)Concept Definition is the set of systems engineering (SE) activities in which the problem space and the needs and requirements of the business or enterprise and stakeholders are closely examined. The activities are grouped and described as generic processes which include Mission Analysis and Stakeholder Needs and Requirements. Concept Definition begins before any formal definition of the system-of-interest (SoI) is developed.

Mission Analysis focuses on the needs and requirements of business or enterprise — that is, on defining the problem or opportunity that exists (in what is often called the problem space or problem situation), as well as understanding the constraints on and boundaries of the selected system when it is fielded (in what is often called the solution space). The Stakeholder Needs and Requirements process explores and defines the operational aspects of a potential solution for the stakeholders from their point of view, independent of any specific solution. In these two Concept Definition activities, business or enterprise decision makers and other stakeholders describe *what* a solution should accomplish and *why* it is needed. Both *why* and *what* need to be answered before consideration is given to *how* the problem will be addressed (i.e., what type of solution will be implemented) and *how* the solution will be defined and developed.

If a new or modified system is needed then System Definition activities are performed to assess the system. See Life Cycle Processes and Enterprise Need for further detail on the transformation of needs and requirements from the business or enterprise and stakeholder levels of abstraction addressed in Concept Definition to the system and system element level of abstraction addressed in System Definition.

The specific activities and sequence of Concept Definition activities and their involvement with the life cycle activities of any system, and in particular the close integration with System Definition activities, will be dependent upon the type of life cycle model being utilized. See Applying Life Cycle Processes for further discussion of the concurrent, iterative and recursive nature of these relationships.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Business or Mission Analysis
- Stakeholder Needs and Requirements

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 as well as topics covered in Part 3.

Concept Definition Activities

There are two primary activities discussed under concept definition: Mission Analysis and the definition of Stakeholder Needs and Requirements:

1. Mission Analysis begins an iteration of the life cycle of a potential SoI that could solve a problem or realize an opportunity for developing a new product, service, or enterprise. These activities assist business or enterprise decision makers to define the problem space, identify the stakeholders, develop preliminary operational concepts, and distinguish environmental conditions and constraints that bound the solution space. In other words, mission analysis takes the enterprise capability gap or opportunity and defines the problem/opportunity in a manner that provides a common understanding encapsulated in what are referred to as “business or mission needs”. Business or mission needs are then used to produce a clear, concise, and verifiable set of business requirements.
2. The Stakeholder Needs and Requirements activity works with stakeholders across the life cycle to elicit and capture a set of needs, expectations, goals, or objectives for a desired solution to the problem or opportunity, referred to as "stakeholder needs". The stakeholder needs are used to produce a clear, concise, and verifiable set of stakeholder requirements. Stakeholder needs and requirements identify and define the needs and requirements of the stakeholders in a manner that enables the characterization of the solution alternatives.

Mission Analysis takes the business and stakeholders' needs and requirements and carries the analysis down from problem space to solution space, including concept, mission, and boundary or context so that a solution concept (at the black-box level) can be selected from the alternatives. Figure 1 in the Mission Analysis topic depicts this interaction. The products and artifacts produced during Concept Definition are then used in System Definition.

The different aspects of how systems thinking is applicable to concept definition are discussed in SEBoK Part 2. In particular, the use of a combination of hard system and soft system approaches depending on the type of problem or class of solution is discussed in Identifying and Understanding Problems and Opportunities and the contrast between top-down and bottom up approaches in Synthesizing Possible Solutions.

Drivers of Solution on Problem Definition: Push Versus Pull

Problem definition and solution design depend on each other. Solutions should be developed to respond appropriately to well-defined problems. Problem definitions should be constrained to what is feasible in the solution space. System Analysis activities are used to provide the link between problems and solutions.

There are two paradigms that drive the ways in which concept definition is done: *push* and *pull*. The *pull* paradigm is based on providing a solution to an identified problem or gap, such as a missing mission capability for defense or infrastructure. The *push* paradigm is based on creating a solution to address a perceived opportunity, such as the emergence of an anticipated product or service that is attractive to some portion of the population (i.e. whether a current market exists or not). This can have an effect on other life cycle processes, such as in verification and validation, or alpha/beta testing as done in some commercial domains.

As systems generally integrate existing and new system elements in a mixture of push and pull, it is often best to combine a bottom-up approach with a top-down approach to take into account legacy elements, as well as to identify the services and capabilities that must be provided in order to define applicable interface requirements and constraints. This is discussed in Applying Life Cycle Processes.

References

Works Cited

None.

Primary References

ANSI/EIA. 1998. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

Additional References

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* Hoboken, NJ, USA: John Wiley & Sons.

ISO/IEC. 2003. Systems Engineering – A Guide for The Application of ISO/IEC 15288 System Life Cycle Processes. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 19760:2003 (E). http://www.hitchins.net/EmergenceEtc.pdf.

ISO/IEC. 2007. *Systems Engineering – Application and Management of The Systems Engineering Process*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* (April 2010): 41-43.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C., USA: National Aeronautics and Space Administration (NASA). NASA/SP-2007-6105.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Business or Mission Analysis

The starting point of engineering any system-of-interest (SoI) is understanding the socio-economic and technological context in which potential problems or opportunities reside. Then, the enterprise strategic goals and stakeholder needs, expectations, and requirements represent the problem or the opportunity from the viewpoint of business or enterprise decision makers while also taking into account the views of users, acquirers, and customers.

Mission Analysis (MA) is part of the larger set of Concept Definition (glossary) activities - the set of systems engineering activities in which the problem space and the needs of the business or enterprise and stakeholders are closely examined; this occurs before any formal definition of the (SoI) is developed, but may need to be revisited through the life cycle. In fact, the activities of Concept Definition determine whether the enterprise strategic goals and business needs will be addressed by a new system, a change to an existing system, a service, an operational change or some other solution. The MA activity focuses on the identification of the primary purpose(s) of the solution (its "mission"), while Stakeholder Needs and Requirements activity explores what capabilities stakeholders desire in accomplishing the mission and may include some detail on the performance of certain aspects of the solution. MA is often performed iteratively with the Stakeholder Needs and Requirements activity to better understand the problem (or opportunity) space, as well as the solution space.

Purpose and Definition

The purpose of MA is to understand a mission/market problem or opportunity, analyze the solution space, and initiate the life cycle of a potential solution that could address the problem or take advantage of an opportunity. MA is a type of strategic or operations analysis related to needs, capability gaps, or opportunities and solutions that can be applied to any organization that evolves its strategy for its business objectives.

MA, in some domains called market analysis or business analysis, is the identification, characterization, and assessment of an operational problem or opportunity within an enterprise. The definition of a mission or business function in a problem space frames the solution, both in terms of the direct application to the mission or business function, and in terms of the context for the resulting solution.

MA is used to define needed (or desired) operational actions, not hardware/software functions; that is, it is focused on defining the problem space, not the solution space. MA begins with the business vision and Concept of Operations (ConOps) (IEEE. 1998), and other organization strategic goals and objectives including the mission (or business function). The primary products of MA are Business or Mission Needs, which are supported by preliminary life-cycle concepts—including a preliminary acquisition concept, a preliminary operational concept (OpsCon), a preliminary deployment concept, a preliminary support concept, and a preliminary retirement concept. Business or Mission Needs are then elaborated and formalized into Business or Mission Requirements. The preliminary operational concept includes the operational scenarios for the mission and the context in which the solution will exist.

MA may include mathematical analysis, modeling, simulation, visualization, and other analytical tools to characterize the intended mission and determine how to best achieve the needs/objectives. MA evaluates alternative approaches to determine which best supports the stakeholder needs (among both materiel and non-materiel solution alternatives, also known as product solutions and service/operational solutions). Thus, MA defines the problem space and analyzes the solution space alternatives using quality attribute constraints driven by the enterprise objectives.

Principles and Concepts

Mission Analysis and Concept of Operations

MA and the terms ConOps and OpsCon are broadly used in U.S. and UK defense and aerospace organizations to analyze and define how a system is intended to operate, as well as how the major operations or operational scenarios are intended to be performed. They take into account the strategic, operational, and tactical aspects of the identified scenarios. ANSI/AIAA G-043A-2012 (ANSI 2012) identifies that the terms 'concept of operations' and 'operational concept' are often used interchangeably but notes that an important distinction exists because each has a separate purpose and is used to meet different ends. The ConOps is at an organisational level, prepared by enterprise management and refined by business management:

> *The ConOps, at the organization level, addresses the leadership's intended way of operating the organization. It may refer to the use of one or more systems (as black boxes) to forward the organization's goals and objectives. The ConOps document describes the organization's assumptions or intent in regard to an overall operation or series of operations within the business in regards to the system to be developed, existing systems, and possible future systems. This document is frequently embodied in long-range strategic plans and annual operational plans. The ConOps document serves as a basis for the organization to direct the overall characteristics of future business and systems.* (ISO/IEC 2011)

The ConOps informs the OpsCon, which is drafted by business management in the Mission Analysis activity and refined by stakeholders in the Stakeholder Needs and Requirements activity:

> *A system OpsCon document describes what the system will do (not how it will do it) and why (rationale). An OpsCon is a user-oriented document that describes system characteristics of the to-be-delivered system from the user's viewpoint. The OpsCon document is used to communicate overall quantitative and qualitative system characteristics to the acquirer, user, supplier and other organizational elements.* (ISO/IEC 2011)

It should be noted that the OpsCon has an operational focus and should be supported by the development of other concepts, including a deployment concept, a support concept, and a retirement concept.

In order to determine appropriate technical solutions for evolving enterprise capabilities, systems engineering (SE) leaders interact with enterprise leaders and operations analysts to understand

- the enterprise ConOps and future mission, business, and operational (MBO) objectives;
- the characterization of the operational concept and objectives (i.e., constraints, mission or operational scenarios, tasks, resources, risks, assumptions, and related missions or operations); and
- how specific missions or operations are currently conducted and what gaps exist in those areas.

They then conceptually explore and select from alternative candidate solutions. This interaction ensures a full understanding of both the problem space and the solution space. The alternative candidate solutions can include a wide range of approaches to address the need, as well as variants for an approach to optimize specific characteristics (e.g., using a different distribution of satellite orbit parameters to maximize coverage or events while minimizing the number of satellites). Analysis, modeling and simulation, and trade studies are employed to select alternative approaches (NDIA 2010).

The notions of mission analysis, ConOps and OpsCon are also used in industrial sectors, such as aviation administrations and aeronautic transportation, health care systems, and space with adapted definitions and/or terms, such as operational concepts, usage concepts and/or technological concepts. For example, “mission analysis” is the term used to describe the mathematical analysis of satellite orbits performed to determine how best to achieve the objectives of a space mission (ESA 2008).

In commercial sectors, MA is often primarily performed as market analysis. Wikipedia defines market analysis as a process that:

> *. . . studies the attractiveness and the dynamics of a special market within a special industry. It is part of the industry analysis and this in turn of the global environmental analysis. Through all these analyses, the chances, strengths, weaknesses, and risks of a company can be identified. Finally, with the help of a Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis, adequate business strategies of a company will be defined. The market analysis is also known as a documented investigation of a market that is used to inform a firm's planning activities, particularly around decisions of inventory, purchase, work force expansion/contraction, facility expansion, purchases of capital equipment, promotional activities, and many other aspects of a company.* (Wikipedia Contributors, 2012)

Anywhere these notions are used, it is evident that they are based on fundamental concepts, such as the operational mode (or state of the system), scenario (of actions), the enterprise level ConOps and the system level operational concepts, functions, etc. For more explanations about the ConOps and operational concept, refer to *Systems and Software Engineering - Requirements Engineering* (ISO/IEC 2011); useful information can be found in Annex A, "System Operational Concept", and Annex B, "Concept of Operations" (ISO/IEC 2011).

Mission Analysis as Part of Enterprise Strategy Development

Periodically, most enterprises re-evaluate their strategy with respect to their mission, vision, and positioning to accomplish their goals. Figure 1 shows the interactions of the enterprise strategy development and the concept definition, including the MA and Stakeholder Needs and Requirements activities that are involved in an iterative manner to fully develop the strategy and define future capabilities and solutions.

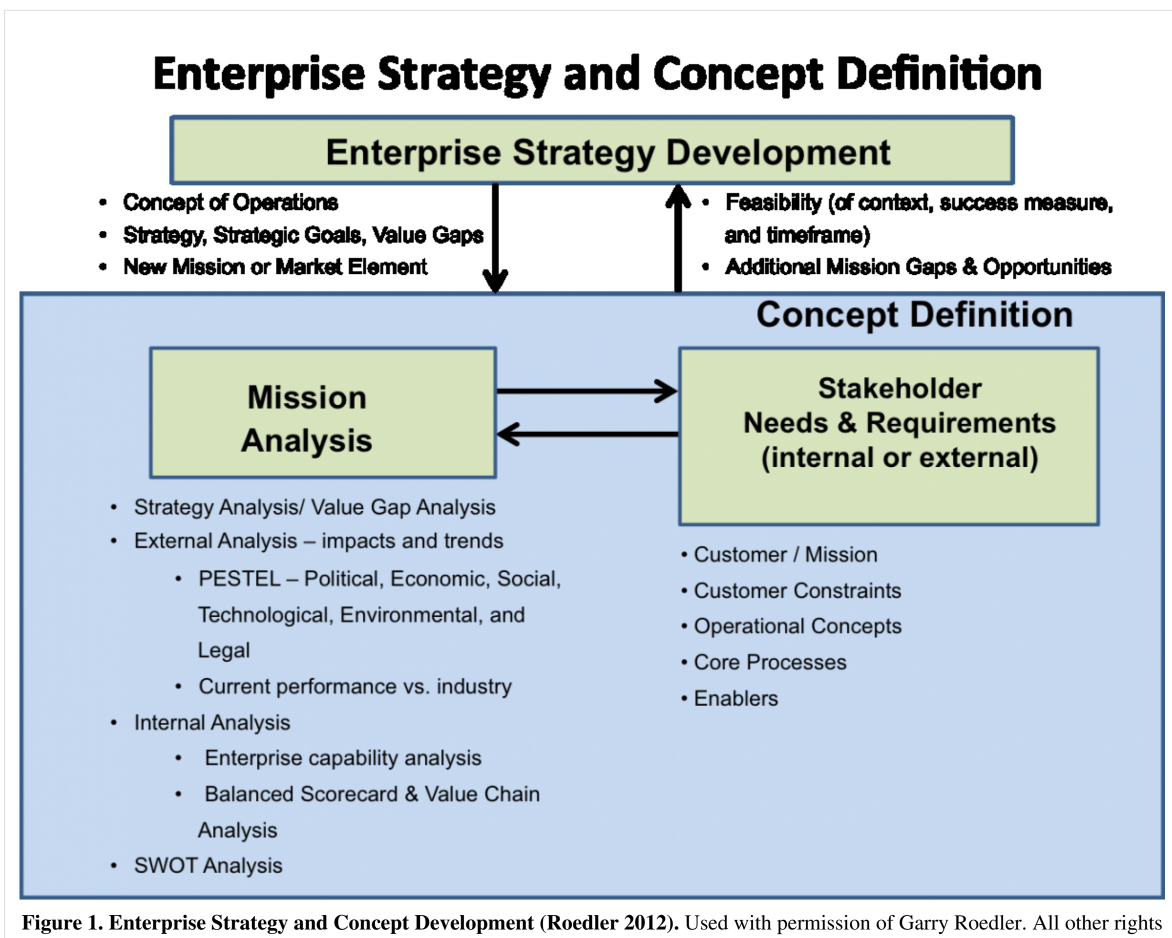


Figure 1. Enterprise Strategy and Concept Development (Roedler 2012). Used with permission of Garry Roedler.

As the enterprise evolves the strategy, it is essential to conduct the supporting MA or strategic analysis for each element of the enterprise to determine readiness to achieve future objectives. This analysis examines the current state to identify any problems or opportunities related to the objective achievement and aids the enterprise in fully understanding and defining the problem space. The analysis examines the external environment and interfaces in search of impacts and trends, as well as the internal enterprise to gauge its capabilities and value stream gaps. Additionally, a strengths, weaknesses, opportunities, and threats (SWOT) analysis may be performed. As the problem space is defined, the stakeholder needs are defined and transformed into stakeholder requirements that define the solutions needed. These requirements include those that address customer and mission needs, the future state of core processes and capabilities of the enterprise, and the enablers to support performance of those processes and capabilities. Finally, MA is engaged again to examine the solution space. Candidate solutions that span the potential solution space are identified, from simple operational changes to various system developments or modifications. Various techniques are applied to analyze the candidates, understand their feasibility and value, and select the best alternative.

Process Approach

Activities of the Process

It is necessary to perform the following major activities and tasks during the MA process:

1. Review and understand the enterprise mission, vision, and ConOps.
2. Identify and define any gaps and opportunities related to future evolution of the strategy:
    1. Examine the current state to identify any problems or opportunities related to the objective achievement, including any deficiencies of the existing system.
    2. Analyze the context of the actual political, economic, social, technological, environmental, and legal (PESTAL) factors, while studying sensitive factors such as cost and effectiveness, security and safety improvement, performance improvement or lack of existing systems, market opportunities, regulation changes, users' dissatisfaction, etc. External, internal, and SWOT analysis should be included as well. For the technological considerations, an appropriate architecture framework representation, such as the U.S. Department of Defense Architecture Framework (DoDAF) operations view (DoD 2010), the Zachman Framework (Rows1 and 2) (Zachman 2008), and The Open Group Architecture Framework (TOGAF) Architecture Development Method (ADM) (The Open Group 2010) Phases A and B should be included within the concept definition when performing mission analysis and stakeholders needs and requirements.
    3. Define the mission, business, and/or operational problem or opportunity, as well as its context, and any key parameters, without focusing on a solution.
3. Examine and evaluate the solution space.
    1. Identify the main stakeholders (customers, users, administrations, regulations, etc.).
    2. Identify high level operational modes or states, or potential use cases.
    3. Identify candidate solutions that span the potential solution space, from simple operational changes to various system developments or modifications. Identify existing systems, products, and services that may address the need for operational or functional modifications. Deduce what potential expected services may be needed. The SoI is a potential and not yet existing product, service or enterprise. Additionally, the solution could be an operational change or a change to an existing product or service.
4. Perform appropriate modeling, simulation, and analytical techniques to understand the feasibility and value of the alternative candidate solutions. Model or simulate operational scenarios from these services and use cases, and enrich them through reviews with stakeholders and subject matter experts.
5. Define basic operational concept or market strategy, and/or business models.

   1. From previous modeled operational scenarios and operational modes, deduce and express the usage of operational concepts, or technical concepts.
   2. Collect and enrich needs, expectations, scenarios, and constraints.
   3. Validate the mission of any potential SoI in the context of any proposed market strategy or business model.
6. Evaluate the set of alternatives and select the best alternative.
   1. Perform a trade study of the alternatives to discriminate between the alternatives.
7. Provide feedback on feasibility, market factors, and alternatives for use in completion of the enterprise strategy and further actions.
8. Define preliminary deployment concept, preliminary support concept, and preliminary retirement concept.

Mission Analysis Artifacts

This process may create several artifacts, such as

- recommendations for revisions to the enterprise ConOps;
- preliminary operational concept document or inputs;
- mission analysis and definition reports (perhaps with recommendations for revisions of the mission);
- a set of business needs
- preliminary life-cycle concepts (preliminary operational concept, preliminary deployment concept, preliminary support concept, and preliminary retirement concept
- system analysis artifacts (e.g., use case diagrams, context diagrams, sequence/activity diagrams, functional flow block diagrams);
- trade study results (alternatives analysis);
- market study/analysis reports; and
- a set of business (or mission) requirements (often captured in a business requirement specification).

Methods and Modeling Techniques

MA uses several techniques, such as

- use case analysis;
- operational analysis;
- functional analysis;
- technical documentation review;
- trade studies;
- modeling;
- simulation;
- prototyping;
- workshops, interviews, and questionnaires;
- market competitive assessments;
- benchmarking; and
- organizational analysis techniques (e.g., strengths, weaknesses, opportunities, threats (SWOT analysis), and product portfolios).

Practical Considerations

Major pitfalls encountered with mission analysis and marketing analysis are presented in Table 1.

Table 1. Major Pitfalls for Mission Analysis. (SEBoK Original)

Pitfall	Description
Wrong level of system addressed	When delineating the boundaries of the SoI and defining the mission and purpose of the system at the very beginning of systems engineering, a classic mistake is to place the system-of-interest at the wrong level of abstraction. The level of abstraction can be too high or too low (sitting respectively in the upper-system or in a sub-system). This is the consequence of the principle stating that a system is always included in a larger system and of confusing the purpose and the mission of the SoI.
Operational modes or scenarios missing	In commercial products or systems, the lack or insufficient description of operational modes and scenarios (how the SoI will be used, in which situations, etc.) is often encountered.

Proven practices with mission analysis and marketing analysis are presented in Table 2.

Table 2. Mission Analysis Proven Practices. (SEBoK Original)

Practice	Description
Models of operational scenarios	Using modeling techniques as indicated in sections above for operational scenarios in any kind of SoI (including commercial systems).
Models of the context	Consider the context of use as a system and force oneself to use modeling techniques for main aspects of the context (functional, behavioral, physical, etc.).

References

Works Cited

ANSI/AIAA G-043-2012e, Guide to the Preparation of Operational Concept Documents.

DoD. 2010. *DoD Architecture Framework*, version 2.02. Arlington, VA: U.S. Department of Defense. Accessed August 29, 2012. Available at: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf.

ESA. 2008. *Mission Analysis: Towards a European Harmonization.* Paris, France: European Space Agency. Accessed August 29, 2012. Available at: http://www.esa.int/esapub/bulletin/bulletin134/bul134b_schoenmaekers.pdf.

IEEE. 1998. *Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, IEEE 1362:1998.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Life CYcle Processes - Requirements Engineering.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 29148:2011.

NDIA. 2010. "Mission Analysis Committee Charter". Website of the National Defense Industrial Association, Systems Engineering Division, Mission Analysis Committee. Accessed August 29, 2012. Available at: http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Committees/Mission%20Analysis%20Committee/Mission%20Analysis%20Committee%20Charter.pdf.

The Open Group. 2011. *TOGAF*, version 9.1. Hogeweg, The Netherlands: Van Haren Publishing. Accessed August 29, 2012. Available at: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g116.

Wikipedia contributors, "Market analysis," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Market_analysis&oldid=508583878 (accessed August 29, 2012).

Zachman, J. 2008. "John Zachman's Concise Definition of The Zachman Framework™." Zachman International Enterprise Architecture. Accessed August 29, 2012. Available at: http://www.zachman.com/about-the-zachman-framework.

Primary References

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Lamsweerde, A. van. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. New York, NY, USA: Wiley.

Additional References

Center for Quality Management. 1993. "Special Issue on Kano's Methods for Understanding Customer Defined Quality." *Center for Quality Management Journal.* 2(4) (Fall 1993).

Faisandier, A. 2012. *Systems Opportunities and Requirements*. Belberaud, France: Sinergy'Com.

Freeman, R. "Chapter 27: Achieving Strategic Aims: Moving Toward a Process Based Military Enterprise," in *Handbook of Military Industrial Engineering.* A.B. Badiru and M.U. Thomas (eds). Boca Raton, FL, USA: Taylor & Francis Group, CRC Press.

IEEE. 1998. *Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, IEEE 1362:1998.

Hull, M.E.C., K. Jackson, A.J.J. Dick. 2010. *Systems Engineering.* 3rd ed. London, UK: Springer.

Kaplan, R.S. and D.P. Norton. 2008. "Developing the Strategy: Vision, Value Gaps, and Analysis," Balanced Scorecard Report. Cambridge, MA, USA: Harvard Business School Publishing, Jan-Feb 2008.

Kano, N. 1984. "Attractive Quality and Must-Be Quality." *Quality JSQC.* 14(2) (October 1984).

Kohda, T., M. Wada, and K. Inoue. 1994. "A Simple Method for Phased Mission Analysis." *Reliability Engineering & System Safety.* 45(3): 299-309.

Marca, D. A. and C. L. McGowan. 1987. "SADT: Structured analysis and design techniques." *Software Engineering*. New York, NY: McGraw-Hill.

MITRE. 2011. "Concept Development." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/concept_development/ [1].

MITRE. 2011. "Requirements Engineering." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/requirements_engineering/ [2].

MITRE. 2011. "Stakeholder Assessment and Management." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/stakeholder_assessment_management.html/ [3].

Shupp, J.K. 2003. "The Mission Analysis Discipline: Bringing focus to the fuzziness about Attaining Good Architectures." Proceedings of INCOSE 13th International Symposium, July 2003.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/concept_development/
[2] http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/requirements_engineering/
[3] http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/stakeholder_assessment_management.html/

Stakeholder Needs and Requirements

Stakeholder needs and requirements represent the views of those at the business or enterprise operations level—that is, of users, acquirers, customers, and other stakeholders as they relate to the problem (or opportunity), as a set of requirements for a solution that can provide the services needed by the stakeholders in a defined environment. Using enterprise-level life cycle concepts (see Business or Mission Analysis for details) as guidance, stakeholders are led through a structured process to elicit stakeholder needs (in the form of a refined set of system-level life-cycle concepts). Stakeholder needs are transformed into a defined set of Stakeholder Requirements, which may be documented in the form of a model, a document containing textual requirement statements or both.

Stakeholder requirements play major roles in systems engineering, as they:

- Form the basis of system requirements activities.
- Form the basis of system validation and stakeholder acceptance .
- Act as a reference for integration and verification activities.
- Serve as means of communication between the technical staff, management, finance department, and the stakeholder community.

This topic describes the definition of stakeholder needs and requirements which involves the activities necessary to elicit and prioritize the needs of the stakeholder(s), and transform those needs into a set of defined stakeholder requirements. Defining the problem or the issue to be solved, identifying the opportunity for developing a new solution, or improving a system-of-interest (SoI) must begin prior to starting the activities necessary to define stakeholder needs and requirements. This means that an initial context of use of the new or modified mission, operation, or capability has already been characterized (see Business or Mission Analysis). System requirements are considered in detail during system definition. None of the above can be considered complete until consistency between the two has been achieved, as demonstrated by traceability, for which a number of iterations may be needed.

Purpose and Definition

The purpose of the Stakeholder Needs and Requirements definition activities are to elicit a set of clear and concise needs related to a new or changed mission for an enterprise (see mission analysis (MA) for information relevant to identifying and defining the mission or operation), and to transform these stakeholder needs into verifiable stakeholder requirements.

Stakeholders may well begin with desires, and expectations that may contain vague, ambiguous statements that are difficult to use for SE activities. Care must be taken to ensure that those desires and expectations are coalesced into a set of clear and concise need statements that are useful as a start point for system definition. These need statements will then need to be further clarified and translated into more *engineering-oriented* language in a set of stakeholder requirements to enable proper architecture definition and requirement activities. As an example, a need or an expectation such as, *to easily* manoeuvre *a car in order to park*, will be transformed in a set of stakeholder requirements to a statement such as, *increase the driviability of the car*, *decrease the effort for handling*, *assist the*

piloting, *protect the coachwork against shocks or scratch*, etc.

To allow a clear description of the activities of stakeholder needs and requirements to be described a generic view of the business teams and roles involved in a typical enterprise has been used below, this includes teams such a business management and business operations; and roles including requirements engineer and business analyst. For an overview of these roles and how they enable both stakeholder and business requirements across the layers of a typical enterprise see Life Cycle Processes and Enterprise Need.

Principles and Concepts

Identifying Stakeholders

Stakeholders of a SoI may vary throughout the life cycle. Thus, in order to get a complete set of needs and subsequent requirements, it is important to consider all stages of the life cycle model when identifying the stakeholders or classes of stakeholders.

Every system has its own stages of life, which typically include stages such as concept, development, production, operations, sustainment, and retirement (for more information, please see Life Cycle Models). For each stage, a list of all stakeholders having an interest in the future system must be identified. The goal is to get every stakeholder's point of view for every stage of the system life in order to consolidate a complete set of stakeholder needs that can be prioritized and transformed into the set of stakeholder requirements as exhaustively as possible. Examples of stakeholders are provided in Table 1.

Table 1. Stakeholder Identification Based on Life Cycle Stages. (SEBoK Original)

Life Cycle Stage	Example of Related Stakeholders
Engineering	Acquirer, panel of potential users, marketing division, research and development department, standardization body, suppliers, verification and validation team, production system, regulator/certification authorities, etc.
Development	Acquirer, suppliers (technical domains for components realization), design engineers, integration team, etc.
Transfer for Production or for Use	Quality control, production system, operators, etc.
Logistics and Maintenance	Supply chain, support services, trainers, etc.
Operation	Normal users, unexpected users, etc.
Disposal	Operators, certifying body, etc.

Identifying Stakeholder Needs

Once business management is satisfied that their needs and requirements are reasonably complete, they pass them on to the business operations team. Here, the Stakeholder Needs and Requirements (SNR) Definition Process uses the ConOps, or Strategic Business Plan (SBP), and the life-cycle concepts as guidance. The requirements engineer (RE) or business analyst (BA) leads stakeholders from the business operations layer through a structured process to elicit stakeholder needs—in the form of a refined OpsCon (or similar document) and other life-cycle concepts. The RE or BA may use a fully or partially structured process to elicit specific needs, as described in models such as user stories, use cases, scenarios, system concepts, and operational concepts.

Identifying Stakeholder Requirements

Stakeholder needs are transformed into a formal set of stakeholder requirements, which are captured as models or documented as textual requirements in and output typically called a Stakeholder Requirement Specification (StRS), Stakeholder Requirement Document (StRD) or similar. That transformation should be guided by a well-defined, repeatable, rigorous, and documented process of requirements analysis. This requirements analysis may involve the use of functional flow diagrams, timeline analysis, N2 Diagrams, design reference missions, modeling and simulations, movies, pictures, states and modes analysis, fault tree analysis, failure modes and effects analysis, and trade studies.

Collecting Stakeholder Needs and Requirements

There are many ways to collect stakeholder needs and requirements. It is recommended that several techniques or methods be considered during elicitation activities to better accommodate the diverse set of sources, including:

- Structured brainstorming workshops
- Interviews and questionnaires
- Technical, operational, and/or strategy documentation review
- Simulations and visualizations
- Prototyping
- Modeling
- Feedback from verification and validation processes,
- Review of the outcomes from the system analysis process (ISO/IEC 2015)
- Quality function deployment (QFD) - can be used during the needs analysis and is a technique for deploying the "voice of the customer". It provides a fast way to translate customer needs into requirements. (Hauser and Clausing 1988)
- Use case diagrams (OMG 2010)
- Activity diagrams (OMG 2010)
- Functional flow block diagrams (Oliver, Kelliher, and Keegan 1997)

From the Capture of Stakeholder Needs to the Definition of Stakeholder Requirements

Several steps are necessary to understand the maturity of stakeholder needs and to understand how to improve upon that maturity. Figure 1 presents the *cycle of needs* as it can be deduced from Professor Shoji Shiba's and Professor Noriaki Kano's works and courses, and is adapted here for systems engineering (SE) purposes.

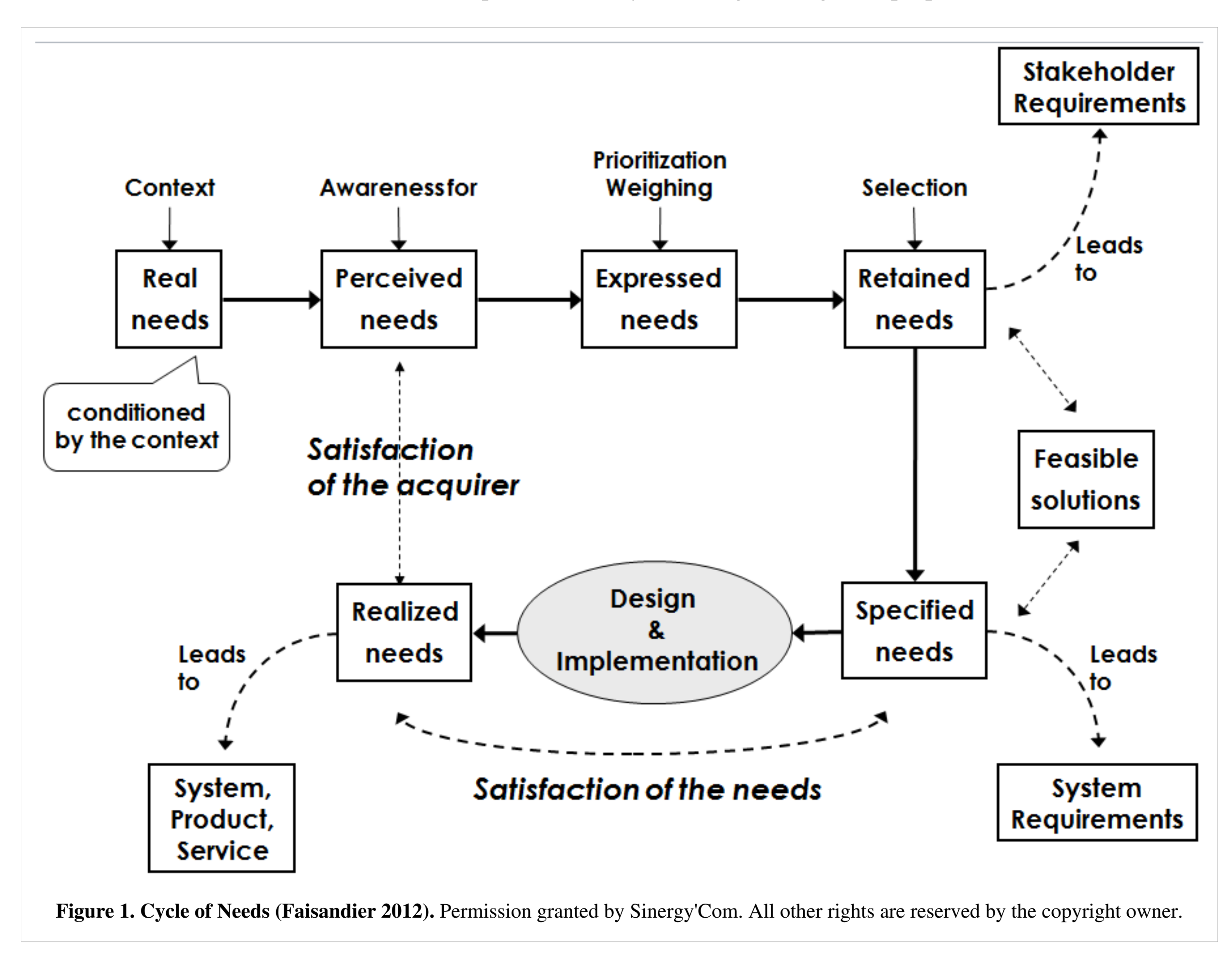


Figure 1. Cycle of Needs (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Figure 1 shows the steps and the position of the stakeholder requirements and system requirements in the engineering cycle. Below are explanations of each stage of requirements (Faisandier 2012); to illustrate this, consider this example of a system related to infectious disease identification:

- **Real needs** are those that lie behind any perceived needs (see below); they are conditioned by the context in which people live. As an example, a generic need could be the ability to *identify infectious diseases easily.*Often, real needs appear to be simple tasks.
- **Perceived needs** are based on a person's awareness that something is wrong, that something is lacking, that improvements could be made, or that there are business, investment, or market opportunities that are not being capitalized upon. Perceived needs are often presented as a list of organized expectations resulting from an analysis of the usage conditions for the considered action (see Business or Mission Analysis). Following from the infectious disease example above, the real need might be perceived as a need to *carry out medical tests in particular circumstances (laboratories, points of care, hospitals, and/or human dispensaries).* Since the real need is seldom clearly expressed, richness of the knowledge of the perceived needs is used as a basis for potential solutions. This step has to be as complete as possible to cover all the contexts of use.
- **Expressed needs** originate from perceived needs in the form of generic actions or constraints, and are typically prioritized. In the example, if safety is the primary concern, the expressed need to *protect the operator against contamination* may take priority over other expressed needs such as *assist in the execution of tests.* When determining the expressed needs, the analysis of the expected mission or services in terms of operational

scenarios takes place.

- **Retained needs** are selected from the expressed needs. The selection process uses the prioritization of expressed needs to achieve a solution or to make attaining solutions feasible. The retained needs allow the consideration of potential solutions for a SoI. These retained *stakeholder intentions do not serve as stakeholder requirements, since they often lack definition, analysis, and possibly consistency and feasibility. Using the concept of operations to aid the understanding of the stakeholder intentions at the organizational level and the system operational concept from the system perspective, requirements engineering leads stakeholders from those initial intentions to structured and more formal stakeholder requirement statements,* ISO/IEC/IEEE 29148 *Systems and software engineering - Requirements engineering* (ISO 2011). Characteristics of good requirements can be found in (ISO 2011). Exploration of potential solutions must start from this step. The various solutions suggested at this step are not yet products, but describe means of satisfying the stakeholder requirements. Each potential solution imposes constraints on the potential future SoI.
- **Specified needs**, are the translation of the stakeholder needs to represent the views of the supplier, keeping in mind the potential, preferred, and feasible solutions. Specified needs are translated into system requirements. Consistent practice has shown this process requires iterative and recursive steps in parallel with other life cycle processes through the system design hierarchy (ISO 2011).
- **Realized needs** are the product, service, or enterprise realized, taking into account every specified need (and hence, the retained needs).

Each class of needs listed above aligns with an area of the SE process. For example, the development of *specified needs* requirements is discussed in the System Requirements topic. For more information on how requirements are used in the systems engineering process, please see the System Definition knowledge area (KA).

Classification of Stakeholder Requirements

Several classifications of stakeholder requirements are possible, e.g. ISO/IEC 29148, section 9.4.2.3 (ISO 2011) provides a useful set of elements for classification. Examples of classification of stakeholder requirements include: service or functional, operational, interface, environmental, human factors, logistical, maintenance, design, production, verification requirements, validation, deployment, training, certification, retirement, regulatory, environmental, reliability, availability, maintainability, design, usability, quality, safety, and security requirements. Stakeholders will also be faced with a number of constraints, including: enterprise, business, project, design, realization, and process constraints.

Process Approach

Activities of the Process

Major activities and tasks performed during this process include the following:

- Identify the stakeholders or classes of stakeholders across the life cycle.
- Elicit, capture, or consolidate the stakeholder needs, expectations, and objectives as well as any constraints coming from the mission and business analysis processes.
- Refine the OpsCon and other life-cycle concepts (acquisition concept, deployment concept, support concept, and retirement concept).
- Prioritize the stakeholder needs.
- Transform the prioritized and retained stakeholder needs into stakeholder requirements.
- Verify the quality of each stakeholder requirement and of the set of stakeholder requirements using the characteristics of good requirements identified in the System Requirements article.
- Validate the content and the relevance of each stakeholder requirement with corresponding stakeholder representatives providing rationale (glossary) for the existence of the requirement.

- Identify potential risks (or threats and hazards) that could be generated by the stakeholder requirements (for further information, see Risk Management).
- Synthesize, record, and manage the stakeholder requirements and potential associated risks.

Artifacts, Methods and Modeling Techniques

This process may create several artifacts, such as:

- Recommendations to refine the Business Requirement Specification (if necessary)
- Refined life-cycle concepts (OpsCon, acquisition concept, deployment concept, support concept, and retirement concept)
- Stakeholder requirements (in the form of a model or a document containing textual requirements, such as the Stakeholder Requirement Specification)
- Stakeholder interview reports
- Stakeholder requirements database
- Stakeholder requirements justification documents (for traceability purposes)
- Input for draft verification and validation plans

The content, format, layout and ownership of these artifacts will vary depending on who is creating them and in which domains they will be used. Between these artifacts and the outputs of the process, activities should cover the information identified in the first part of this article.

Practical Considerations

Major pitfalls encountered with stakeholder requirements are presented in Table 3.

Table 3. Major Pitfalls for Stakeholder Requirements. (SEBoK Original)

Pitfall	Description
Operator Role Not Considered	Sometimes engineers do not take into account the humans acting as operators inside a system or those who use the system and are outside of the system. As a consequence, elements are forgotten (e.g. roles of operators).
Exchanges with External Objects Forgotten	The exhaustiveness of requirements can be an issue; in particular, the interfaces with external objects of the context of the system can be forgotten (exchanges of matter, energy, information).
Physical Connections with External Objects Forgotten	Within the interface issue, physical connections of the system-of-interest with external objects can be forgotten (technological constraints).
Forgotten Stakeholders	Stakeholders can be forgotten, as everyone thinks of direct users, customers, and suppliers; however, one may fail to consider those who do not want the system to exist and malevolent persons.

Proven practices with stakeholder requirements are presented in Table 4.

Table 4. Stakeholder Requirements Proven Practices. (SEBoK Original)

Practice	Description
Involve Stakeholders	Involve the stakeholders early in the stakeholder requirements development process.
Presence of Rationale	Capture the rationale for each stakeholder requirement.
Analyze Sources before Starting	Complete stakeholder requirements as much as possible before starting the definition of the system requirements.
Modeling Techniques	Use modeling techniques as indicated in sections above.
Requirements Management Tool	Consider using a requirements management tool. This tool should have the capability to trace linkages between the stakeholder requirements and the system requirements and to record the source of each stakeholder requirement.

References

Works Cited

Faisandier, A. 2012. *Systems Architecture and Design.* Belberaud, France: Sinergy'Com.

Hauser, J. and D. Clausing. 1988. "The House of Quality." *Harvard Business Review.* (May - June 1988).

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2. Needham, MA: Object Management Group. July 2010.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering complex systems with models and objects*. New York, NY, USA: McGraw-Hill.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements engineering.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Primary References

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Additional References

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

MITRE. 2011. "Requirements Engineering." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/requirements_engineering/.

MITRE. 2011. "Stakeholder Assessment and Management." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/stakeholder_assessment_management.html.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: System Definition

System Definition

System definition activities are conducted to create and describe in detail a system-of-interest (SoI) to satisfy an identified need. The activities are grouped and described as generic processes. which consist of system requirements definition, system architecture definition, system design definition and system analysis. The architecture definition of the system may include the development of related logical architecture models and physical architecture models. During and/or at the end of any iteration, gap analysis is performed to ensure that all system requirements have been mapped to the architecture and design.

System definition activities build on the artifacts and decisions from concept definition, primarily the articulation of the mission of the (SoI), the needs and requirements of stakeholders, and preliminary operational concepts. See Life Cycle Processes and Enterprise Need for further detail on the transformation of needs and requirements from the business or enterprise and stakeholder levels of abstraction addressed in concept definition to the system and system element level of abstraction addressed in system definition.

The products of system definition activities (system requirements, architecture and design) are inputs to system realization.

The specific activities and sequence of system definition activities and their involvement with the life cycle activities of any system, and in particular the close integration with concept definition and system realization activities, will be dependent upon the type of life cycle model being utilized. See Applying Life Cycle Processes for further discussion of the concurrent, iterative and recursive nature of these relationships.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Requirements
- System Architecture
- Logical Architecture Model Development
- Physical Architecture Model Development
- System Design
- System Analysis

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

System Views and System Elements

An Engineered System (glossary) solution to a defined concept includes a set of engineering elements, characteristics, and properties. These elements are grouped in two ways:

- Needs and requirements views
- Architecture and design views

Architecture views include the identification of the boundary and interfaces of a system-of-interest (SoI), which may then be further refined as a collection of system elements and their relationships.

Needs and Requirements Views

Requirements provide an overall view of the purpose and mission which the system as a whole is intended to satisfy, as well as a technology-independent view of that the system solutions(s) should do. They are conventionally organized into two types:

- Business or mission requirements and Stakeholder requirements are defined and discussed in the Concept Definition KA.
- System requirements, which describe the functions which the system as a whole should fulfill in order to satisfy the stakeholder requirements and are expressed in an appropriate set of views, and non-functional requirements expressing the levels of safety, security, reliability, etc., which are called for. These collectively form the basis for verification later in the life cycle.

System requirements and stakeholder requirements are closely related. Neither can be considered complete until consistency between the two has been achieved, as demonstrated by traceability, for which a number of iterations may be needed.

The process activities that are used to identify, engineer and manage system requirements are described further in the System Requirements article in the KA.

Architecture and Design Views

A given engineered system is one solution that could address/answer a problem or an opportunity (represented through requirements views); the solution may be more or less complex. A complex solution cannot be comprehended with a single view or model, because of the characteristics or properties of the problem/solution (see system complexity). The characteristics are structured as types or entities; types are related to each other. An instantiation of the set of types can be understood as THE architecture of the system. The majority of interpretations of system architecture are based on the fairly intangible notion of structure. Therefore, the system architecture and design is formally represented with sets of types or entities such as functions, interfaces, resource flow items, information elements, physical elements, nodes, links, etc. These entities may possess attributes/characteristics such as dimensions, environmental resilience, availability, reliability, learnability, execution efficiency, etc. The entities are interrelated by the means of relationships and are generally grouped into sets to represent views/models of the system architecture and design.

Viewpoints and views are sometimes specified in architecture frameworks. Views are usually generated from models. Many systems engineering practices use logical and physical views for modeling the system architecture and design.

- The **logical view of the architecture** supports the logical operation of the system all along its life cycle, and may include functional, behavioral, and temporal views/models. Operational scenarios refine the mission into a collection of functions and dynamic structures that describe how the mission is performed (behavior).
- The **physical view of the architecture** is a set of system elements performing the functions of the system. Those system elements can be either material or immaterial (e.g., equipment made of hardware, software and/or human

roles).

The boundary of the system architecture depends on what engineers include within the scope of the SoI and outside of it. This decision marks the transition from the characterization of the problem context to the beginnings of solution definition.

Facing the potential number of system elements that constitute the physical architecture, sets of system elements can be grouped to form systems. The decomposition of the SoI (highest level) may include the decomposition of several layers of systems (intermediate levels of systems) until technological system elements (lowest level) are defined. Any layer of the decomposition may include systems and non-decomposable technological system elements. The relationship between each layer is recursive; as a system element is also an engineered system it can be characterized in its turn using the previous views in its own context.

The logical and physical representations of the system architecture are mapped onto each other. The interactions between system elements are defined by interfaces whose complexity strongly depends on the way the system architecture and design is defined. The relationships between the outputs of concept definition and the system solution, as well as the range of other views of a system that are available to describe a more complete set of characteristics between the system elements are discussed further in the Logical Architecture Model Development and Physical Architecture Model Development sections of system definition.

System Synthesis and Decomposition

System definition is managed through methodical synthesis of the SoI into systems and system elements. Solution synthesis may be top down or bottom up, as discussed in Synthesizing Possible Solutions. However it is done, as the system architecture definition advances, a decomposition of systems and system elements emerges, this forms a system breakdown structure (SBS). For project management purposes, every system of the SBS may be included in a *building block*, a notion introduced in (ANSI/EIA 1998), also called *system blocks*.

Stakeholder requirements and system requirements exist at all layers of the SBS. In ISO/IEC/IEEE 29148 *Systems and software engineering - Requirements Engineering* (ISO 2011), these layers are known as levels of abstraction. Along with systematically introducing layers of systems, the architecture and design process manages the transformation of the system requirements through levels of abstraction. Figure 1 illustrates this approach.

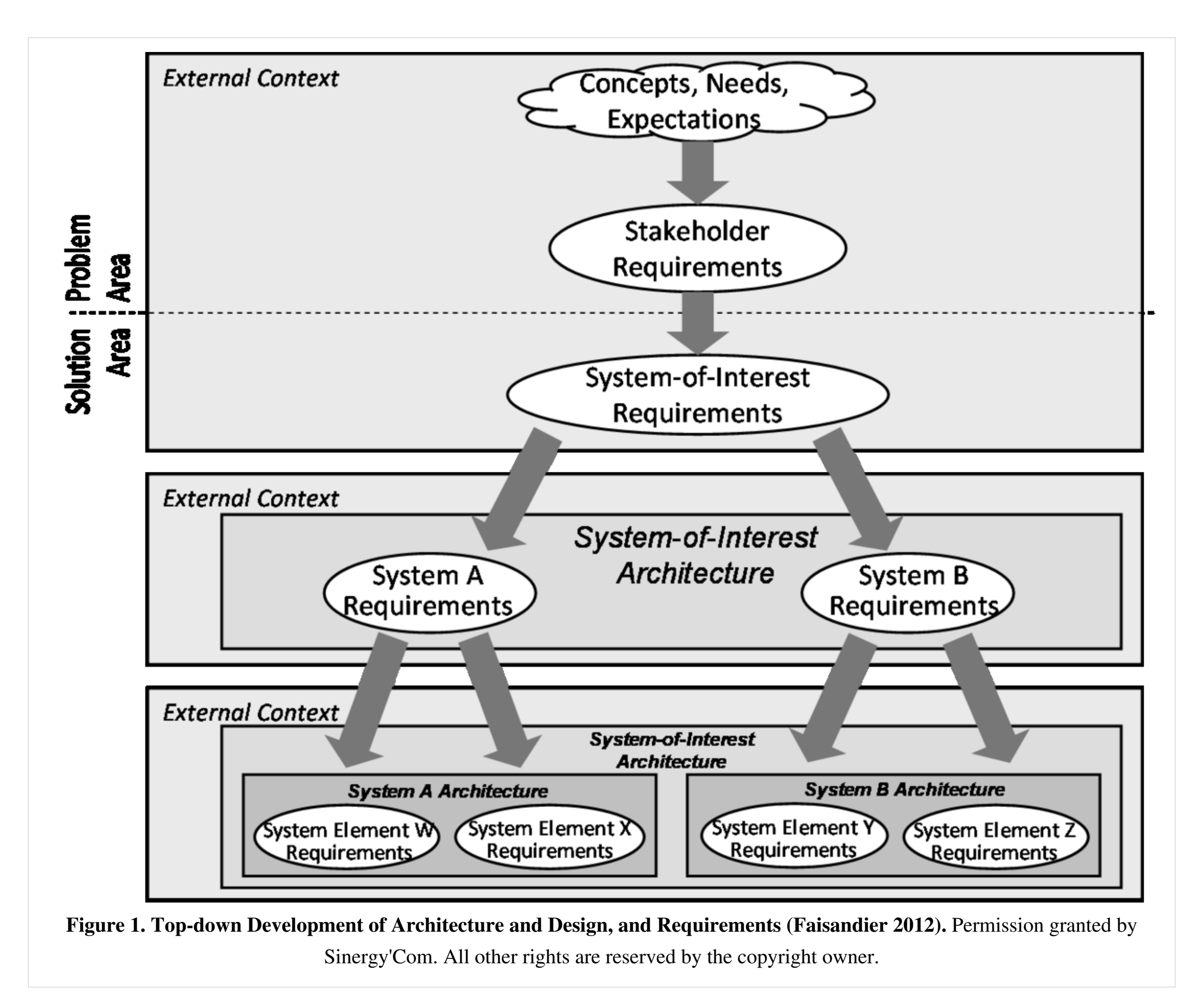


Figure 1. Top-down Development of Architecture and Design, and Requirements (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

As shown in Figure 1

- The white ovals represent requirements at decreasing levels of abstraction, and the arrows represent the transformation of those requirements through the levels using the architecture and design process. Stakeholder expressions of needs, expectations, and constraints are transformed into stakeholder requirements.
- The next transformation crosses the boundary between the problem and solution areas by converting stakeholder requirements into system requirements, reflecting the bounded solution space.
- At the SoI level, the system architecture is developed which serves to identify systems and system elements and establishes how they operate together to address the SoI requirements.

This approach is applied recursively for each level of abstraction/decomposition recognizing that the same generic processes are applied at multiple levels of abstraction. At any level of this decomposition one or more solution options may be presented as system architectures. The process by which the solution which best fits the system requirements, associated stakeholder needs and wider life cycle concerns is selected and justified is discussed in the System Analysis process.

Figure 2 below portrays the engineering that occurs in each system block. As necessary, system elements are defined through sets of system element requirements, which become inputs to other system blocks (*level n+1*). The approach is then recursively applied using the system definition processes.

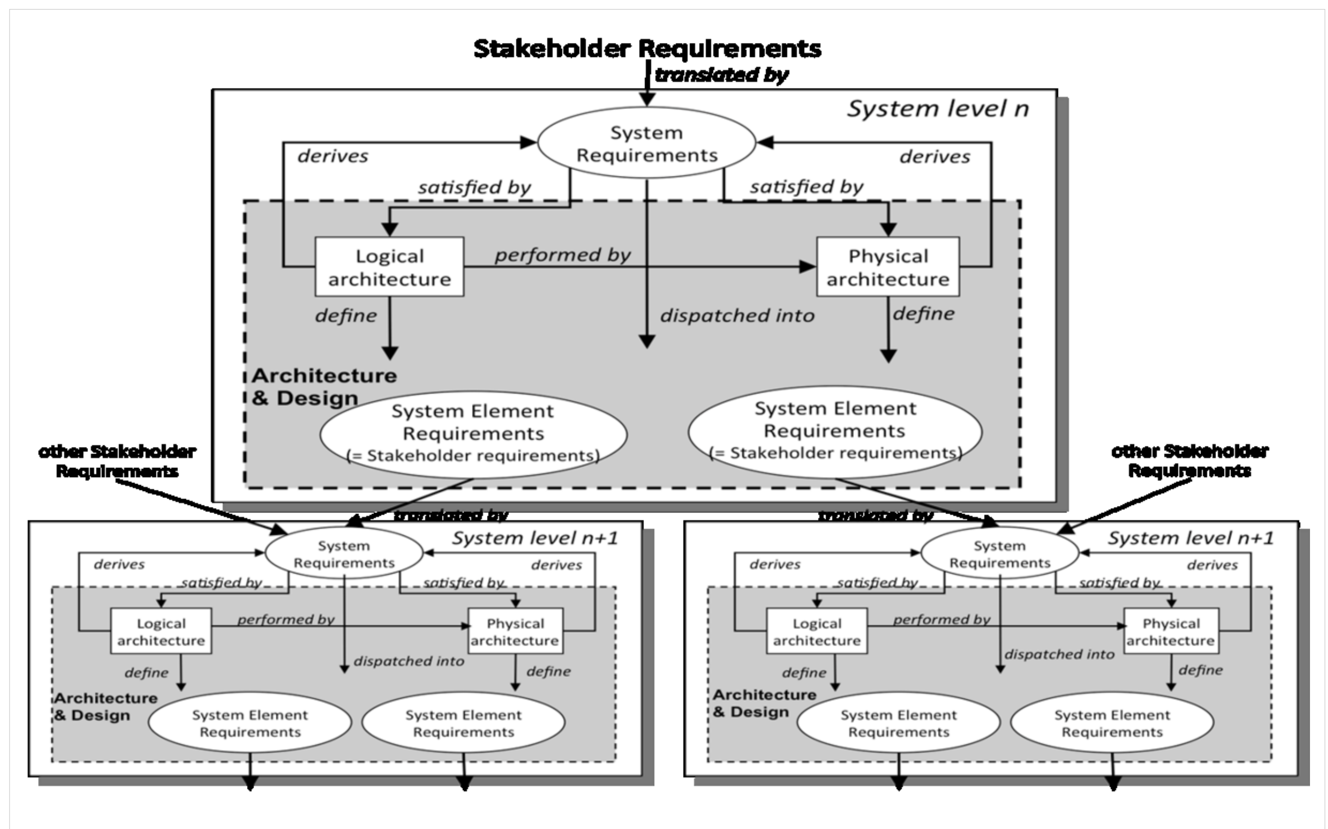


Figure 2. Recursive Instantiation of Definition Processes (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

At the *n+1* level, the systems or system elements may also collect other stakeholder requirements that are directly pertinent to this level of architecture and design. Processes within each system are generic, but unique in local purpose, scope and context.

See Applying Life Cycle Processes for a discussion of the iterative and recursive application of system requirements and architecture processes, and Life Cycle Processes and Enterprise Need for further detail on the transformation of needs and requirements to system and system element levels of abstraction.

The different aspects of how systems thinking is applicable to system definition are discussed in SEBoK Part 2. In particular, see discussion of the recursive nature of systems and engineered system contexts in Engineered System Context; the contrast between top-down and bottom up approaches in Synthesizing Possible Solutions and the role of solution architecture options and selection in Analysis and Selection between Alternative Solutions.

References

Works Cited

ANSI/EIA. 1998. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA-632-1998.

Faisandier, A. 2012. *Systems Architecture and Design.* Belberaud, France: Sinergy'Com.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Primary References

ANSI/EIA. 1998. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis.* 4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC. 2007. *Systems Engineering – Application and Management of The Systems Engineering Process*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products,* 1st ed. Boca Raton, FL, USA: CRC Press.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

Baldwin, C.Y. and K.B. Clark. 2000. *Design Rules*. Cambridge, Mass: MIT Press.

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Hatley, D.J., and I.A. Pirbhai. 1987. *Strategies for Real-Time System Specification*. New York, NY: Dorset House Pub.

MOD. 2010. *MOD Architecture Framework,* Version 1.2.004. UK Ministry of Defence. Available at: http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/InformationManagement/MODAF/.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Requirements

System requirements are all of the requirements at the *system level* that describe the functions which the system as a whole should fulfill to satisfy the stakeholder needs and requirements, and is expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary.

System requirements play major roles in systems engineering, as they:

- Form the basis of system architecture and design activities.
- Form the basis of system Integration (glossary) and verification activities.
- Act as reference for validation and stakeholder acceptance.
- Provide a means of communication between the various technical staff that interact throughout the project.

Elicitation of stakeholder requirements starts in Concept Definition, and will be initially developed though interview and mission analysis. System requirements are considered in detail during System Definition. Neither can be considered complete until consistency between the two has been achieved, as demonstrated by traceability, for which a number of iterations may be needed.

Definition and Purpose of Requirements

A requirement is a statement that identifies a product or processes operational, functional, or design characteristic or constraint, which is unambiguous, testable, or measurable and necessary for product or process acceptability (ISO 2007).

To avoid confusion in the multitude of terms pertaining to *requirements*, consider the following classifications:

- **Process Role or State**: The role the requirement plays in the definition process; for instance, its position in the system block (e.g. translated, derived, satisfied) or its state of agreement (e.g. proposed, approved, cancelled).
- **Level of Abstraction**: The level within the definition process that the requirement stands; for instance, stakeholder requirement, system requirement, system element requirement.
- **Type of Requirement**: The nature of the requirement itself; for instance, functional, performance, constraint, etc.

Any single requirement may simultaneously be in a particular state, at a particular level abstraction, and of a particular type. For additional explanations about differences between the types of requirements, refer to (Martin 1997, Chapter 2).

Principles Governing System Requirements

Relationship to Stakeholder Requirements and Logical Architecture

A set of stakeholder requirements are clarified and translated from statements of need into *engineering-oriented* language in order to enable proper architecture definition, design, and verification activities that are needed as the basis for system requirements analysis.

The system requirements are based around identification and synthesis of the functions required of any solution system associated with performance and other quality measures and provide the basis for the assessment of candidate solutions and verification of the completed system. The system requirements are expressed in technical language that is useful for architecture and design: unambiguous, consistent, coherent, exhaustive, and verifiable. Of course, close coordination with the stakeholders is necessary to ensure the translation is accurate and traceability is maintained. This results in a set of system functions and requirements specifying measurable characteristics which can form the basis for system realization.

The Logical Architecture (glossary) defines system boundary and functions, from which more detailed system requirements can be derived. The starting point for this process may be to identify functional requirements from the stakeholder requirements and to use this to start the architectural definition, or to begin with a high level functional architecture view and use this as the basis for structuring system requirements. The exact approach taken will often depend on whether the system is an evolution of an already understood product or service, or a new and unprecedented solution (see Synthesizing Possible Solutions). However, when the process is initiated it is important that the stakeholder requirements, system requirements, and logical architecture are all complete, consistent with each other, and assessed together at the appropriate points in the systems Life Cycle Model (glossary).

Traceability and the Assignment of System Requirements during Architecture and Design

Requirements traceability provides the ability to track information from the origin of the stakeholder requirements, to the top level of requirements and other system definition elements at all levels of the system hierarchy (see Applying Life Cycle Processes). Traceability is also used to provide an understanding as to the extent of a change as an input when impact analyses is performed in cases of proposed engineering improvements or requests for change.

During architecture definition and design, the assignment of requirements from one level to lower levels in the system hierarchy can be accomplished using several methods, as appropriate - see Table 1.

Table 1. Assessment Types for a System Requirement. (SEBoK Original)

Assignment Type for a System Requirement	Description
Direct Assignment	The system requirement from the higher level is directly assigned to a system or a system element for a lower level (e.g. the color used to paint visible parts of the product).
Indirect Assignment (Simply Decomposed)	The system requirement is distributed across several systems or system elements and the sum of a more complex calculation for distribution is equal to the requirement of higher level (e.g. a mass requirement, power distribution, reliability allocation, etc.) with sufficient margin or tolerance. A documented and configuration-managed "assignment budget" for each assignment must be maintained.
Indirect Assignment (Modeled and Decomposed)	The system requirement is distributed to several systems or system elements using an analysis or mathematical modeling technique. The resulting design parameters are assigned to the appropriate systems or system elements (with appropriate margin). For example, in the case of a radar detection requirement that is being analyzed, these lower-level parameters for output power, beam size, frequencies, etc. will be assigned to the appropriate hardware and software elements. Again, the analysis (or model) must be documented and configuration-managed.
Derived Requirement (from Design)	Such system requirements are developed during the design activities as a result of the decision of the design team, not the stakeholder community. These requirements may include the use of commercial-off-the-shelf (COTS) items, existing systems or system elements in inventory, common components, and similar design decisions in order to produce a "best value" solution for the customer. As such, these derived requirements may not directly trace to a stakeholder requirement, but they do not conflict with a stakeholder requirement or a constraint.

Classification of System Requirements

Several classifications of system requirements are possible, depending on the requirements definition methods and/or the architecture and design methods being applied. (ISO 2011) provides a classification which is summarized in Table 2 (see references for additional classifications).

Table 2. Example of System Requirements Classification. (SEBoK Original)

Types of System Requirement	Description
Functional Requirements	Describe qualitatively the system functions or tasks to be performed in operation.
Performance Requirements	Define quantitatively the extent, or how well, and under what conditions a function or task is to be performed (e.g. rates, velocities). These are quantitative requirements of system performance and are verifiable individually. Note that there may be more than one performance requirement associated with a single function, functional requirement, or task.
Usability Requirements	Define the quality of system use (e.g. measurable effectiveness, efficiency, and satisfaction criteria).
Interface Requirements	Define how the system is required to interact or to exchange material, energy, or information with external systems (external interface), or how system elements within the system, including human elements, interact with each other (internal interface). Interface requirements include physical connections (physical interfaces) with external systems or internal system elements supporting interactions or exchanges.
Operational Requirements	Define the operational conditions or properties that are required for the system to operate or exist. This type of requirement includes: human factors, ergonomics, availability, maintainability, reliability, and security.
Modes and/or States Requirements	Define the various operational modes of the system in use and events conducting to transitions of modes.
Adaptability Requirements	Define potential extension, growth, or scalability during the life of the system.
Physical Constraints	Define constraints on weight, volume, and dimension applicable to the system elements that compose the system.
Design Constraints	Define the limits on the options that are available to a designer of a solution by imposing immovable boundaries and limits (e.g., the system shall incorporate a legacy or provided system element, or certain data shall be maintained in an online repository).
Environmental Conditions	Define the environmental conditions to be encountered by the system in its different operational modes. This should address the natural environment (e.g. wind, rain, temperature, fauna, salt, dust, radiation, etc.), induced and/or self-induced environmental effects (e.g. motion, shock, noise, electromagnetism, thermal, etc.), and threats to societal environment (e.g. legal, political, economic, social, business, etc.).
Logistical Requirements	Define the logistical conditions needed by the continuous utilization of the system. These requirements include sustainment (provision of facilities, level support, support personnel, spare parts, training, technical documentation, etc.), packaging, handling, shipping, transportation.
Policies and Regulations	Define relevant and applicable organizational policies or regulatory requirements that could affect the operation or performance of the system (e.g. labor policies, reports to regulatory agony, health or safety criteria, etc.).
Cost and Schedule Constraints	Define, for example, the cost of a single exemplar of the system, the expected delivery date of the first exemplar, etc.

Requirements Management

Requirements management is performed to ensure alignment of the system and system element requirements with other representations, analysis, and artifacts of the system. It includes providing an understanding of the requirements, obtaining commitment, managing changes, maintaining bi-directional traceability among the requirements and with the rest of the system definition, and alignment with project resources and schedule.

There are many tools available to provide a supporting infrastructure for requirements management; the best choice is the one that matches the processes of the project or enterprise. Requirements management is also closely tied to configuration management for baseline management and control. When the requirements have been defined, documented, and approved, they need to be put under baseline management and control. The baseline allows the project to analyze and understand the impact (technical, cost, and schedule) of ongoing proposed changes.

Process Approach

Purpose and Principle of the Approach

The purpose of the system requirements analysis process is to transform the stakeholder, user-oriented view of desired services and properties into a technical view of the product that meets the operational needs of the user. This process builds a representation of the system that will meet stakeholder requirements and that, as far as constraints permit, does not imply any specific implementation. It results in measurable system requirements that specify, from the supplier's perspective, what performance and non-performance characteristics it must possess in order to satisfy stakeholders' requirements (ISO 2015).

Activities of the Process

Major activities and tasks during this process include:

1. Analyzing the stakeholder requirements to check completeness of expected services and operational scenarios, conditions, operational modes, and constraints.
2. Defining the system requirements and their rationale.
3. Classifying the system requirements using suggested classifications (see examples above).
4. Incorporating the derived requirements (coming from architecture and design) into the system requirements baseline.
5. Establishing the upward traceability with the stakeholder needs and requirements.
6. Establishing bi-directional traceability between requirements at adjacent levels of the system hierarchy.
7. Verifying the quality and completeness of each system requirement and the consistency of the set of system requirements.
8. Validating the content and relevance of each system requirement against the set of stakeholder requirements.
9. Identifying potential risks (or threats and hazards) that could be generated by the system requirements.
10. Synthesizing, recording, and managing the system requirements and potential associated risks.
11. Upon approval of the requirements, establishing control baselines along with the other system definition elements in conjunction with established configuration management practices.

Checking Correctness of System Requirements

System requirements should be checked to gauge whether they are well expressed and appropriate. There are a number of characteristics that can be used to check system requirements, such as standard peer review techniques and comparison of each requirement against the set of requirements characteristics, which are listed in Table 2 and Table 3 of the "Presentation and Quality of Requirements" section (below). Requirements can be further validated using the requirements elicitation and rationale capture described in the section "Methods and Modeling Techniques" (below).

Methods and Modeling Techniques

Requirements Elicitation and Prototyping

Requirements elicitation requires user involvement and can be effective in gaining stakeholder involvement and buy-in. Quality Function Deployment (QFD) and prototyping are two common techniques that can be applied and are defined in this section. In addition, interviews, focus groups, and Delphi techniques are often applied to elicit requirements.

QFD is a powerful technique to elicit requirements and compare design characteristics against user needs (Hauser and Clausing 1988). The inputs to the QFD application are user needs and operational concepts, so it is essential that the users participate. Users from across the life cycle should be included to ensure that all aspects of user needs are accounted for and prioritized.

Early prototyping can help the users and developers interactively identify functional and operational requirements as well as user interface constraints. This enables realistic user interaction, discovery, and feedback, as well as some sensitivity analysis. This improves the users' understanding of the requirements and increases the probability of satisfying their actual needs.

Capturing Requirements Rationale

One powerful and cost-effective technique to translate stakeholder requirements to system requirements is to capture the rationale for each requirement. Requirements rationale is merely a statement as to why the requirement exists, any assumptions made, the results of related design studies, or any other related supporting information. This supports further requirements analysis and decomposition. The rationale can be captured directly in a requirements database (Hull, Jackson, and Dick 2010).

Some of the benefits of this approach include:

- **Reducing the total number of requirements** - The process aids in identifying duplicates. Reducing requirements count will reduce project cost and risk.
- **Early exposure of bad assumptions**
- **Removes design implementation** - Many poorly written stakeholder requirements are design requirements in disguise, in that the customer is intentionally or unintentionally specifying a candidate implementation.
- **Improves communication with the stakeholder community** - By capturing the requirements rationale for all stakeholder requirements, the line of communication between the users and the designers is greatly improved. (Adapted from Chapter 8 of (Hooks and Farry 2000)).

Modeling Techniques

Modeling techniques that can be used when requirements must be detailed or refined, or in cases in which they address topics not considered during the stakeholder requirements definition and mission analysis, include:

- State-charts models (ISO 2011, Section 8.4)
- Scenarios modeling (ISO 2011, Section 6.2.3.1)
- Simulations, prototyping (ISO 2011, Section 6.3.3.2)
- Quality Function Deployment (INCOSE 2011, p. 83)
- Systems Modeling Language (SysML) sequence diagrams, activity diagrams, use cases, state machine diagrams, requirements diagrams (OMG 2010)
- Functional Flow Block Diagram for operational scenarios (Oliver, Kelliher, and Keegan 1997)

Presentation and Quality of Requirements

Generally, requirements are provided in a textual form. Guidelines exist for writing good requirements; they include recommendations about the syntax of requirements statements, wording (exclusions, representation of concepts, etc.), and characteristics (specific, measurable, achievable, feasible, testable, etc.). Refer to (INCOSE 2011, Section 4.2.2.2) and (ISO 2011).

There are several characteristics of both requirements and sets of requirements that are used to aid their development and to verify the implementation of requirements into the solution. Table 3 provides a list and descriptions of the characteristics for individual requirements and Table 4 provides a list and descriptions of characteristics for a set of requirements, as adapted from (ISO 2011, Sections 5.2.5 and 5.2.6).

Table 3. Characteristics of Individual Requirements. (SEBoK Original)

Characteristic	Description
Necessary	The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is not included in the set of requirements, a deficiency in capability or characteristic will exist, which cannot be fulfilled by implementing other requirements
Appropriate	The specific intent and amount of detail of the requirement is appropriate to the level of the entity to which it refers (level of abstraction). This includes avoiding unnecessary constraints on the architecture or design to help ensure implementation independence to the extent possible
Unambiguous	The requirement is stated in such a way so that it can be interpreted in only one way.
Complete	The requirement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the entity need without needing other information to understand the requirement
Singular	The requirement should state a single capability, characteristic, constraint, or quality factor.
Feasible	The requirement can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk
Verifiable	The requirement is structured and worded such that its realization can be proven (verified) to the customer's satisfaction at the level the requirements exists.
Correct	The requirement must be an accurate representation of the entity need from which it was transformed.
Conforming	The individual requirements should conform to an approved standard template and style for writing requirements, when applicable.

Note: Traceability is considered by some sources as a characteristic (ISO 2011). However, a recent viewpoint is that Traceability is actually an attribute of a requirement; that is, something that is appended to the requirement, not an intrinsic characteristic of a requirement (INCOSE 2011). The traceability characteristic or attribute is defined as: The requirement is upwards traceable to specific documented stakeholder statement(s) of need, higher tier requirement, or another source (e.g., a trade or design study). The requirement is also downwards traceable to the specific requirements in the lower tier requirements specifications or other system definition artifacts. That is, all parent-child relationships for the requirement are identified in tracing such that the requirement traces to its source and implementation.

Table 4. Characteristics of a Set of Requirements. (SEBoK Original)

Characteristic	Description
Complete	The requirement set stands alone such that it sufficiently describes the necessary capabilities, characteristics, constraints, and/or quality factors to meet the entity needs without needing other information. In addition, the set does not contain any to be defined (TBD), to be specified (TBS), or to be resolved (TBR) clauses.
Consistent	The set of requirements contains individual requirements that are unique, do not conflict with or overlap with other requirements in the set, and the units and measurement systems they use are homogeneous. The language used within the set of requirements is consistent, i.e., the same word is used throughout the set to mean the same thing.
Feasible	The requirement set can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk. (Note: Feasible includes the concept of "affordable".)
Comprehensible	The set of requirements must be written such that it is clear as to what is expected by the entity and its relation to the system of which it is a part.
Able to be validated	It must be able to be proven the requirement set will lead to the achievement of the entity needs within the constraints (such as cost, schedule, technical, legal and regulatory compliance).

Requirements in Tables

Requirements may be provided in a table, especially when specifying a set of parameters for the system or a system element. It is good practice to make standard table templates available. For tables, the following conventions apply:

- Invoke each requirements table in the requirements set that clearly points to the table.
- Identify each table with a unique title and table number.
- Include the word “requirements” in the table title.
- Identify the purpose of the table in the text immediately preceding it and include an explanation of how to read and use the table, including context and units.
- For independent-dependent variable situations, organize the table in a way that best accommodates the use of the information.
- Each cell should contain, at most, a single requirement.

Requirements in Flow Charts

Flow charts often contain requirements in a graphical form. These requirements may include logic that must be incorporated into the system, operational requirements, process or procedural requirements, or other situations that are best defined graphically by a sequence of interrelated steps. For flow charts, the following conventions apply:

- Invoke flow charts in the requirements set that clearly points to the flow chart.
- Identify each flow chart with a unique title and figure number.
- Include the word “requirements” in the title of the flow chart.
- Clearly indicate and explain unique symbols that represent requirements in the flow chart.

Requirements in Drawings

Drawings also provide a graphical means to define requirements. The type of requirement defined in a drawing depends on the type of drawing. The following conventions apply:

- Drawings are used when they can aid in the description of the following:
  - Spatial Requirements
  - Interface Requirements
  - Layout Requirements
- Invoke drawings in the requirements set that clearly point to the drawing.

Artifacts

This process may create several artifacts, such as:

- System Requirements Document
- System Requirements Justification Document (for traceability purpose)
- System Requirements Database, including traceability, analysis, rationale, decisions, and attributes, where appropriate.
- System External Interface Requirements Document (this document describes the interfaces of the system with external elements of its context of use; the interface requirements can be integrated or not to the system requirements document.

The content, format, layout and ownership of these artifacts will vary depending on who is creating them as well as in which domain they will be utilized. Between them and the outputs of the process, activities should cover the information identified in the first part of this article.

Practical Considerations about System Requirements

There are several **pitfalls** that will inhibit the generation and management of an optimal set of system requirements, as discussed in Table 5.

Table 5. Major Pitfalls with Definition of System Requirements. (SEBoK Original)

Pitfall	Description
Insufficient Analysis of Stakeholder Requirements	If the receivers of the stakeholder requirements do not perform a sufficient critical analysis of them, the consequence could be difficulties translating them into system requirements and the obligation to come back to the stakeholders, losing time.
Insufficient Analysis of Operational Modes and Scenarios	The operational modes and operational scenarios are not sufficiently analyzed or defined by the person in charge of writing the system requirements. Those elements allow the structuring of the system and its use early in the engineering process and help the designer to remember functions and interfaces.
Incomplete Set of System Requirements	If the system requirements are not sufficiently precise and complete, there is a great risk that the design will not have the expected level of quality and that the verification and validation of the system will be delayed.
Lack of Verification Method	Delaying the capture of verification methods and events for each system requirement; identification of the verification approach for each requirement often provides additional insight as to the correctness and necessity of the requirement itself.
Missing traceability	Incorrect or missing traceability of each requirement, both to an upper-level "parent" requirement as well as allocation to an inappropriate system or system element.

The **proven practices** in Table 6 have repeatedly been shown to reduce project risk and cost, foster customer satisfaction, and produce successful system development.

Table 6. Proven Practices for System Requirements. (SEBoK Original)

Practice	Description
Involve Stakeholders	Involve the stakeholders as early as possible in the system requirements development process.
Presence of Rationale	Capture the rationale for each system requirement.
Always Complete before Starting	Check that stakeholder requirements are complete as much as possible before starting the definition of the system requirements.
Peer Reviews	Organize peer reviews of system requirements with applicable subject matter experts.
Modeling Techniques	Use modeling techniques as indicated in sections above.
Requirements Management Tool	Consider using a requirements management tool, especially for more complex projects. This tool should have the capability to trace linkages between system requirements to display relationships. A requirements management tool is intended to facilitate and support the systematic managing of system requirements throughout the project life cycle.
Measures for Requirement Engineering	Use typical measures for requirement engineering; for further information, refer to the *Systems Engineering Leading Indicators Guide* (Roedler et al. 2010). Both process and product measures should be used for requirements engineering. To get the desired insight to facilitate risk-managed requirements engineering, it may be necessary to use more than one measure based on the information needs (risks, objectives, issues) for the requirements. Useful measures include: • Requirements Volatility • Requirements Trends • Requirements Verification Progress (plan vs. actual) • Requirements Validation Progress (plan vs. actual) • TBD and TBR Closure Per Plan • Peer Review Defects

References

Works Cited

Hauser, J. and D. Clausing. 1988. "The House of Quality." *Harvard Business Review.* (May - June 1988).

Hooks, I.F. and K.A. Farry. 2000. *Customer-centered products: Creating successful products through smart requirements management*. New York, NY, USA: American Management Association.

Hull, M.E.C., K. Jackson, A.J.J. Dick. 2010. *Systems Engineering,* 3rd ed. London, UK: Springer.

INCOSE. 2011. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

ISO/IEC. 2007. *Systems and Software Engineering -- Recommended Practice for Architectural Description of Software-Intensive Systems*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 42010:2007.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Martin, J.N. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products,* 1st ed. Boca Raton, FL, USA: CRC Press.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering complex systems with models and objects*. New York, NY, USA: McGraw-Hill.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2. Needham, MA: Object Management Group. July 2010.

Primary References

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Lamsweerde, A. van. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. New York, NY, USA: Wiley.

Additional References

Faisandier, A. 2012. *Systems Opportunities and Requirements*. Belberaud, France: Sinergy'Com.

Hooks, I.F. and K.A. Farry. 2000. *Customer-Centered Products: Creating Successful Products through Smart Requirements Management.* New York, NY, USA: American Management Association.

Hull, M.E.C., K. Jackson, A.J.J. Dick. 2010. *Systems Engineering,* 3rd ed. London, UK: Springer.

Roedler, G., D. Rhodes, C. Jones, and H. Schimmoller. 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2005-001-03.

SEI. 2007. "Requirements Management Process Area" and "Requirements Development Process Area." in *Capability Maturity Model Integrated (CMMI) for Development*, version 1.2. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Architecture

The purpose of system architecture activities is to define a comprehensive solution based on principles, concepts, and properties logically related and consistent with each other. The solution architecture has features, properties, and characteristics satisfying, as far as possible, the problem or opportunity expressed by a set of system requirements (traceable to mission/business and stakeholder requirements) and life cycle concepts (e.g., operational, support) and are implementable through technologies (e.g., mechanics, electronics, hydraulics, software, services, procedures, human activity).

System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high-level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. It may also be applied to more than one system, in some cases forming the common structure, pattern, and set of requirements for classes or families of similar or related systems.

General Concepts and Principles

Notion of Structure

The SEBoK considers systems engineering to cover all aspects of the creation of a system, including system architecture.

The majority of interpretations of system architecture are based on the fairly intangible notion of *structure* (i.e. relationships between elements). Some authors limit the types of structure considered to be architectural; for example, restricting themselves to *functional* and *physical* structure. Recent practice has extended consideration to include *behavioral*, *temporal* and other dimensions of structure.

ISO/IEC/IEEE 42010 *Systems and software engineering - Architecture description* (ISO 2011) provides a useful description of the architecture considering the stakeholder concerns, architecture viewpoints, architecture views, architecture models, architecture descriptions, and architecting throughout the life cycle.

A discussion of the features of systems architectures can be found in (Maier and Rechtin 2009).

An attempt to develop and apply a systematic approach to characterizing architecture belief systems in systems engineering has been described by the INCOSE UK Architecture Working Group (Wilkinson et al.2010, Wilkinson 2010).

Architecture Description of the System

An architecture framework contains standardized viewpoints, view templates, meta-models, model templates, etc. that facilitate the development of the views of a system architecture (see Architecture Framework (glossary) for examples). ISO/IEC/IEEE 42010 (ISO 2011) specifies the normative features of architecture frameworks, viewpoints, and views as they pertain to architecture description. A viewpoint addresses a particular stakeholder concern (or set of closely related concerns). The viewpoint specifies the kinds of model to be used in developing the system architecture to address that concern (or set of concerns), the ways in which the models should be generated, and how the models are related and used to compose a view.

Logical and physical models (or views) are often used for representing fundamental aspects of the system architecture. Other complementary viewpoints and views are necessarily used to represent how the system architecture addresses stakeholder concerns, for example, cost models, process models, rule models, ontological models, belief models, project models, capability models, data models, etc.

Classification of Principles and Heuristics

Engineers and architects use a mixture of mathematical principles and heuristics (heuristics are lessons learned through experience, but not mathematically proven). When an issue is identified and defined through system requirements, principles and heuristics may or may not be able to address it. Principles and heuristics that are used in system views/models can be classified according to the domains in which those system views/models are used, as follows:

1. **Static domain** relates to physical structure or organization of the SoI broken down into systems and system elements. It deals with partitioning systems, system elements, and physical interfaces.
2. **Dynamic domain** relates to logical architecture models; in particular, to the representation of the behavior of the system. It includes a description of functions (i.e. transformations of input flows into output flows) and interactions between functions of the system and between those of the external objects or systems. It takes into account reactions to events that launch or stop the execution of functions of the system. It also deals with the effectiveness (i.e. performances, operational conditions) of the system.
3. **Temporal domain** relates to temporal invariance levels of the execution of functions of the system. This means that every function is executed according to cyclic or synchronous characteristics. It includes decisional levels that are asynchronous characteristics of the behavior of some functions.
4. **Environmental domain** relates to enablers (production, logistics support, etc.), but also to the survivability of the system in reaction to natural hazards or threats and to the integrity of the system in reaction to internal potential hazards. This includes, for example, climatic, mechanical, electromagnetic, and biological aspects.

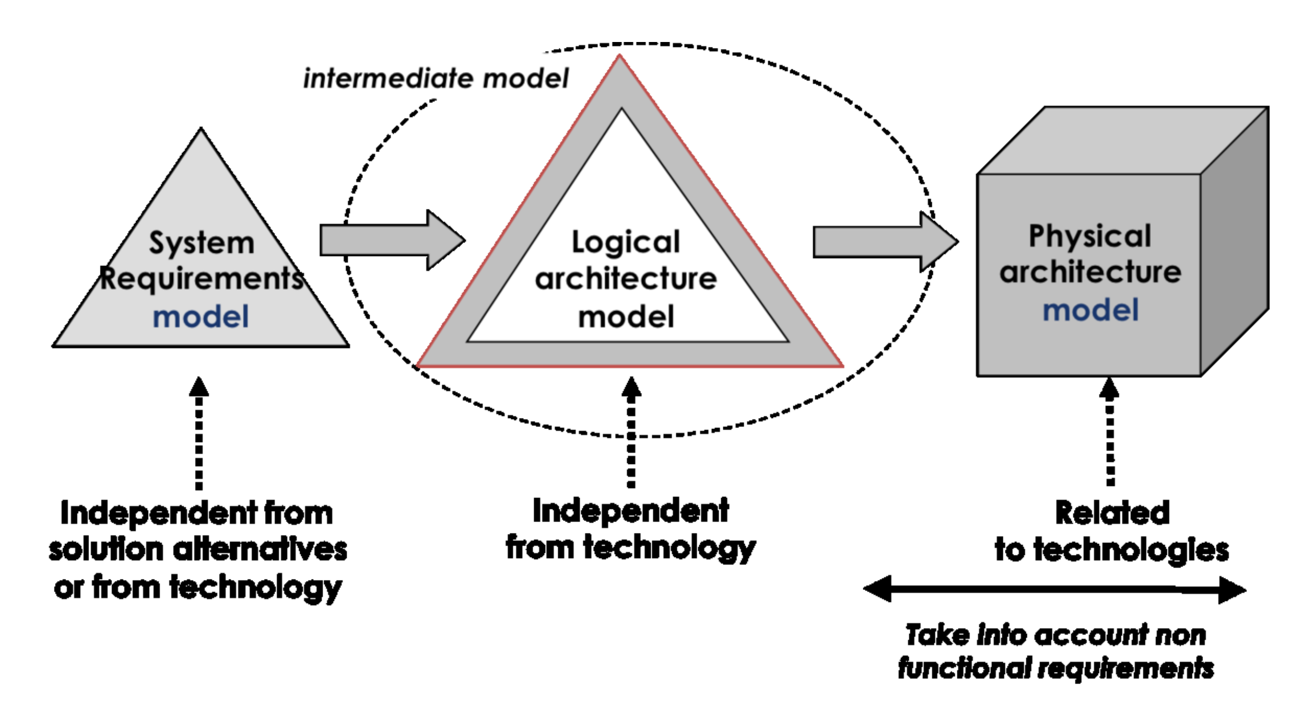
More detailed classification of heuristics can be found in (Maier and Rechtin 2009).

Transition from System Requirements to Logical and Physical Architecture Models

The aim of the approach is to progress from system requirements (representing the problem from a supplier/designer point of view, as independent of technology as possible) through an intermediate model of logical architecture, to allocate the elements of the logical architecture model to system elements of candidate physical architecture models.

(System requirements and logical architecture models share many characteristics, as they are both organized on functional lines, independently of the implementation. Some authors (Stevens et al 1998) go so far as to conflate the two, which simplifies the handling of multiple simultaneous views. Whether this approach is adopted depends on the specific practices of the development organization and where contractual boundaries are drawn.)

Design decisions and technological solutions are selected according to performance criteria and non-functional requirements, such as operational conditions and life cycle constraints (e.g., environmental conditions, maintenance constraints, realization constraints, etc.), as illustrated in Figure 1. Creating intermediate models, such as logical architecture models, facilitates the validation of functional, behavioral, and temporal properties of the system against the system requirements that have no major technological influence impacts during the life of the system, the physical interfaces, or the technological layer without completely questioning the logical functioning of the system.



Iterations between Logical and Physical Architecture Model Development

As discussed in system requirements, the exact approach taken in the synthesis of solutions will often depend on whether the system is an evolution of an already understood product or service, or a new and unprecedented solution (see Synthesizing Possible Solutions).

Whatever the approach, architecture activities require spending several iterations between logical architecture models development and physical architecture models development, until both logical and physical architecture models are consistent and provide the necessary level of detail. One of the first architecture activities is the creation of a logical architecture model based on nominal scenarios (of functions). The physical architecture model is used to determine main system elements that could perform system functions and to organize them.

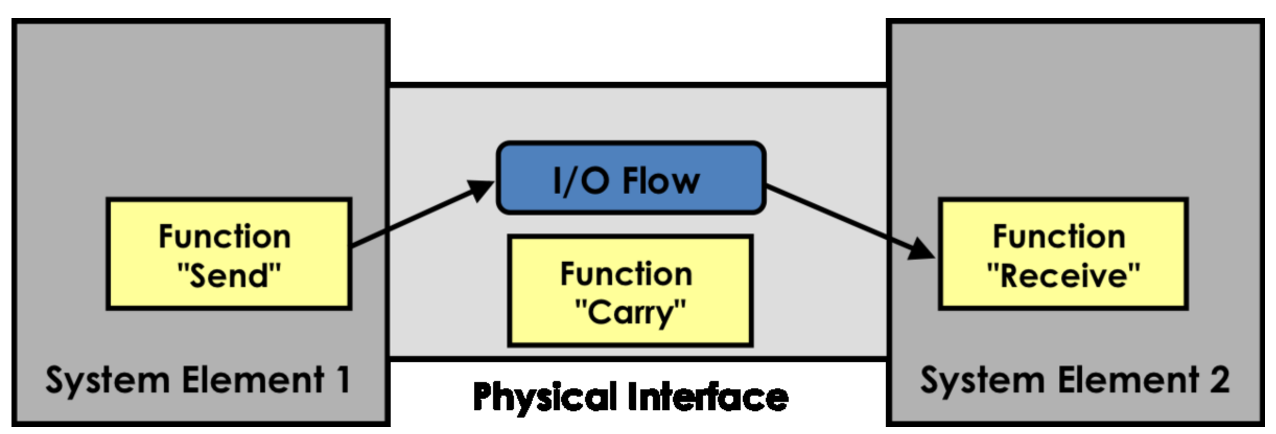
Subsequent logical architecture model iterations can take into account allocations of functions to system elements and derived functions coming from physical solution choices. It also supplements the initial logical architecture model by introducing other scenarios, failure analyses, and operational requirements not previously considered. Derived functions are allocated to system elements; in turn, this affects the physical architecture models.

Additional iterations are focused on producing complete and consistent logical and physical views of the solution.

During system design, technological choices can potentially lead to new functions, new input/output and control flows, and new physical interfaces. These new elements can lead to creation of new system requirements, called *derived requirements*.

Notion of Interface

The notion of interface is one of the most important to consider when defining the architecture of a system. The fundamental aspect of an interface is functional and is defined as inputs and outputs of functions. As functions are performed by physical elements (system elements), inputs/outputs of functions are also carried by physical elements; these are called physical interfaces. Consequentially, both functional and physical aspects are considered in the notion of interface. A detailed analysis of an interface shows the function *“send”* located in one system element, the function *“receive”* located in the other one, and the function *“carry"* as being performed by the physical interface that supports the input/output flow (see Figure 2).



In the context of complex exchanges between system elements, particularly in software-intensive systems, a protocol is seen as a physical interface that carries exchanges of data. However, the input/output flows can include many other exchanges than data, such as energy.

Emergent Properties

The overarching architecture of a system may have design properties or operational effects that emerge from the arrangement and interaction between system elements, but which may not be properties of any individual element or intended for the system as a whole.

The elements of an engineered system interact among themselves and can create desirable or undesirable phenomena, such as inhibition, interference, resonance, or the reinforcement of any property. The definition of the system includes an analysis of interactions between system elements in order to prevent undesirable properties and reinforce desirable ones.

A property which emerges from a system can have various origins, from a single system element to the interactions among several elements (Thome, B. 1993). The term emergent properties is used by some authors to identify any property which emerges from a system, while other may refer to this as synergy and reserve emergent property for explaining unexpected properties or properties not considered fully during system development, but have emerged during operation. The system concept of emergence is discussed in SEBoK Part 2 (see Emergence).

Broad Categories of Properties	Description and Examples
Local Property	The property is located in a single system element – e.g. the capacity of a container is the capacity of the system.
Accumulative System Property	The property is located in several system elements and is obtained through the simple summation of elemental properties – e.g. the weight of the system results from the sum of the weights of its system elements.
Emergent Property Modified by Architecture and/or Interactions.	The property exists in several system elements and is modified by their interactions – e.g. the reliability/safety of a system results from the reliability/safety of each system element and the way they are organized. Architectural steps are often critical to meeting system requirements.
Emergent Property Created by Interactions	The property does not exist in system elements and results only from their interactions – e.g. electromechanical interfaces, electromagnetism, static electricity, etc.
Controlled Emergent Property	Property controlled or inhibited before going outside the system – e.g.: unbalance removed by the addition of a load; vibration deadened by a damper.

Physical architecture design will include the identification of likely synergies and emergent properties and the inclusion of derived functions, components, arrangements, and/or environmental constraints in the logical or physical architectures models to avoid, mitigate or restrain them within acceptable limits. Corresponding *derived requirements* should be added to the system requirements baseline when they impact the system-of-interest(SoI). This may be achieved through the knowledge and experience of the systems engineer or through the application of system patterns. However, it is generally not possible to predict, avoid, or control all emergent properties during

the architecture development. Fully dealing with the consequences of emergence can only be done via iteration between system definition, system realization and system deployment and use (Hitchins, 2008)

The notion of emergence is applied during architecture and design to highlight necessary derived functions; additionally, internal emergence is often linked to the notion of complexity. This is the case with complex adaptive systems (CAS), in which the individual elements act independently, but behave jointly according to common constraints and goals (Flood and Carson 1993). Examples of CAS include: the global macroeconomic network within a country or group of countries, stock market, complex web of cross border holding companies, manufacturing businesses, geopolitical organizations, etc. (Holland, J. 1999 and 2006).

Reuse of System Elements

Systems engineers frequently utilize existing system elements. This reuse constraint has to be identified as a system requirement and carefully taken into account during architecture and design. One can distinguish three general cases involving system element reuse, as shown in Table 2.

Re-use Case	Actions and Comments
Case 1: The requirements of the system element are up-to-date and it will be re-used with no modification required.	• The system architecture to be defined will have to adapt to the boundaries, interfaces, functions, effectiveness, and behavior of the re-used system element. • If the system element is not adapted, it is probable that costs, complexity, and risks will increase.
Case 2: The requirements of the system element are up-to-date and it will be re-used with possible modifications.	• The system architecture to be defined is flexible enough to accommodate the boundaries, interfaces, functions, effectiveness, and behavior of the re-used system element. • The design of the reused system element, including its test reports and other documentation, will be evaluated and potentially redesigned.
Case 3: The requirements are not up-to-date or do not exist.	• It is necessary to reverse engineer the system element to identify its boundaries, interfaces, functions, performances, and behavior. • This is a difficult activity, since the extant documentation for the re-used system element is likely unavailable or insufficient. • Reverse engineering is expensive in terms of both time and money, and brings with it increased risk.

There is a common idea that reuse is *free*; however, if not approached correctly, reuse may introduce risks that can be significant for the project (costs, deadlines, complexity).

Process Approach

Purpose

The purpose of the System Architecture process is to generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views. (ISO 2015).

It should be noted that the architecture activities below overlap with both system definition and concept definition activities. In particular that key aspects of the operational and business context, and hence certain stakeholder needs, strongly influence the approach taken to architecture development and description. Also, the architecture activities will drive the selection of, and fit within, whatever approach to solution synthesis has been selected.

Activities of the process

Major activities and tasks performed during this process include the following:

1. Initialize the definition of the system architecture

- Build an understanding of the environment/context of use for which a system is needed in order to establish insight into the stakeholder concerns. To do this, analyze relevant market, industry, stakeholder, enterprise, business, operations, mission, legal and other information that help to understand the perspectives that could guide the definition of the system architecture views and models.
- Capture stakeholder concerns (i.e., expectations or constraints) that span system life cycle stages. The concerns are often related to critical characteristics to the system that relate to the stages; they should be translated into or incorporated to system requirements.
- Tag system requirements that deal with operational conditions (e.g., safety, security, dependability, human factors, interfaces, environmental conditions) and life cycle constraints (e.g., maintenance, disposal, deployment) that would influence the definition of the architecture elements.
- Establish an architecture roadmap and strategy that should include methods, modeling techniques, tools, need for any enabling systems, products, or services, process requirements (e.g., measurement approach and methods), evaluation process (e.g., reviews and criteria).
- Plan enabling products or services acquisition (need, requirements, procurement).

2. Define necessary architecture viewpoints

- Based on the identified stakeholder concerns, identify relevant architecture viewpoints and architecture frameworks that may support the development of models and views.

3. Develop candidate architectures models and views

- Using relevant modeling techniques and tools, and in conjunction with the Stakeholder Needs and Requirements process and the System Requirements process, determine the system-of-interest context including boundary with elements of the external environment. This task includes the identification of relationships, interfaces or connections, exchanges and interactions of the system-of-interest with external elements. This task enables to define or understand the expected operational scenarios and/or system behaviors within its context of use.
- Define architectural entities (e.g., functions, input/output flows, system elements, physical interfaces, architectural characteristics, information/data elements, containers, nodes, links, communication resources, etc.), which address the different types of system requirements (e.g., functional requirements, interface requirements, environmental requirements, operational conditions – dependability, human factors, etc., constraints – physical dimensions, production, maintenance, disposal).
- Relate architectural entities to concepts, properties, characteristics, behaviors, functions, and/or constraints that are relevant to decisions of the system-of-interest architecture. This gives rise to architectural characteristics (e.g., generality, modularity, operability, efficiency, simplicity).
- Select, adapt, or develop models of the candidate architectures of the system, such as logical and physical models (see **Logical Architecture Model Development** and **Physical Architecture Model Development**). It is sometimes neither necessary nor sufficient to use logical and physical models. The models to be used are those that best address key stakeholder concerns.
- From the models of the candidate architectures, compose views that are relevant to the stakeholder concerns and critical or important requirements.
- Define derived system requirements induced by necessary instances of architectural entities (e.g., functions, interfaces) and by structural dispositions (e.g., constraints, operational conditions). Use the system requirements definition process to define and formalize them.

- Check models and views consistency and resolve any identified issues. ISO/IEC/IEEE 42010, 2011 maybe used for this.
- Verify and validate the models by execution or simulation, if modeling techniques and tools permit. Where possible, use design tools to check feasibility and validity; and/or implement partial mock-ups, or use executable architecture prototypes or simulators.

4. Relate system architecture to system design

- Define the system elements that reflect the architectural characteristics (when the architecture is intended to be design-agnostic, these system elements may be notional until the design evolves). To do this, partition, align, and allocate architectural characteristics and system requirements to system elements. Establish guiding principles for the system design and evolution. Sometimes, a “reference architecture” is created using these notional system elements as a means to convey architectural intent and to check for design feasibility.
- Define interfaces for those that are necessary for the level of detail and understanding of the architecture. This includes the internal interfaces between the system elements and the external interfaces with other systems.
- Determine the design properties applicable to system elements in order to satisfy the architectural characteristics.
- For each system element that composes the system, develop requirements corresponding to allocation, alignment, and partitioning of design properties and system requirements to system elements. To do this, use the stakeholder needs and requirements definition process and the system requirements definition process.

5. Assess architecture candidates and select one

- Assess the candidate architectures using the architecture evaluation criteria. This is done through application of the System Analysis, Measurement, and Risk Management processes.
- Select the preferred architecture(s). This is done through application of the Decision Management process.

6. Manage the selected architecture

- Establish and maintain the rationale for all selections among alternatives and decision for the architecture, architecture framework(s), viewpoints, kinds of models, and models of the architecture.
- Manage the maintenance and evolution of the architecture description, including the models, and views. This includes concordance, completeness, and changes due to environment or context changes, technological, implementation, and operational experiences. Allocation and traceability matrices are used to analyze impacts onto the architecture. The present process is performed at any time evolutions of the system occur.
- Establish a means for the governance of the architecture. Governance includes the roles, responsibilities, authorities, and other control functions.
- Coordinate reviews of the architecture to achieve stakeholder agreement. The stakeholder requirements and system requirements can serve as references.

Artifacts, Methods and Modeling Techniques

This process may create several artifacts, such as system architecture description documents and system justification documents (traceability matrices and architectural choices).

The content, format, layout, and ownership of these artifacts may vary depending on the person creating them and the domains in which they are being used. The outputs of the process activities should cover the information identified in the first part of this article.

Practical Considerations

Pitfalls

Some of the key pitfalls encountered in planning and performing system architecture are provided in Table 3.

Pitfall	Description
Problem Relevance	If the architecture is developed without input from the stakeholders' concerns, or cannot be understood and related back to their issues it might lose the investments of the stakeholder community.
Reuse of System Elements	In some projects, for industrial purposes, existing products or services are imposed very early as architecture/design constraints in the stakeholder requirements or in the system requirements, without paying sufficient attention to the new context of use of the system in which they are also included. It is better to work in the right direction from the beginning. Define the system first, taking note of other requirements, and then see if any suitable non-developmental items (NDI) are available. Do not impose a system element from the beginning, which would reduce the trade-space. The right reuse process consists of defining reusable system elements in every context of use.

Proven Practices

Some proven practices gathered from the references are provided in Table 4.

Practice	Description
Emerging properties	Control the emergent properties of the interactions between the systems or the system elements; obtain the required synergistic properties and control or avoid the undesirable behaviors (vibration, noise, instability, resonance, etc.).

References

Works Cited

Faisandier, A. 2012. *Systems Architecture and Design.* Belberaud, France: Sinergy'Com.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes.* Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting.* 3rd ed. Boca Raton, FL, USA: CRC Press.

Wilkinson, M., A. James, M. Emes, P. King, P. Bryant. 2010. "Belief Systems in Systems Architecting: Method and Preliminary Applications." Presented at the IEEE SMC Society's 5th International Conference on System of Systems Engineering (SoSE). 22nd-24th June 2010. Loughborough University, UK.

Flood, R.L., and E.R. Carson. 1993. *Dealing with complexity: An Introduction to the Theory and Application of Systems Science*, 2nd ed. New York, NY, USA: Plenum Press

Holland, J.H. 1999. *Emergence: from chaos to order*. Reading, Mass: Perseus Books.

Hitchins, D. 2008. "Emergence, Hierarchy, Complexity, Architecture: How do they all fit together? A guide for seekers after enlightenment." Self-published white paper. Accessed 4 September 2012. Available at: http://www.hitchins.net/EmergenceEtc.pdf.

Holland, J.H. 2006. "Studying Complex Adaptive Systems." *Journal of Systems Science and Complexity.* 19(1): 1-8. http://hdl.handle.net/2027.42/41486

Thome, B. 1993. *Systems Engineering, Principles & Practice of Computer-Based Systems Engineering*. New York, NY, USA: Wiley.

Primary References

ANSI/IEEE. 2000. *Recommended Practice for Architectural Description for Software-Intensive Systems*. New York, NY, USA: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Faisandier, A. 2012. *Systems Architecture and Design.*Belberaud, France: Sinergy'Com.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis.*4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

ISO/IEC. 2007. *Systems Engineering – Application and Management of The Systems Engineering Process*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products,*1st ed. Boca Raton, FL, USA: CRC Press.

NASA. 2007. *Systems Engineering Handbook.*Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

Checkland, P. B. 1999. *Systems Thinking, Systems Practice*. Chichester, UK: John Wiley & Sons Ltd.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2, July 2010. http://www.omg.org/technology/documents/spec_catalog.htm.

Sillitto H, "Architecting Systems - Concepts, Principles and Practice", College Publications 2014

Wilkinson, M.K. 2010. "Z8: Systems Architecture", in Z-guide series. INCOSE UK, available from INCOSE UK at: http://www.incoseonline.org.uk/Program_Files/Publications/zGuides.aspx?CatID=Publications.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Logical Architecture Model Development

Logical Architecture Model Development may be used as a task of the activity "Develop candidate architectures models and views", or a sub-process of the System Architecture Definition process (see **System Architecture**). Its purpose is to elaborate models and views of the functionality and behavior of the future engineered system as it should operate, while in service. The logical architecture model of a engineered system of interest (SoI) is composed of a set of related technical concepts and principles that support the logical operation of the system. It may include a functional architecture view, a behavioral architecture view, and a temporal architecture view. Other additional views are suggested in architecture frameworks, depending on the domain.

Note: The term *Logical Architecture* is a contraction of the expression *Logical View of the System Architecture*.

Concepts and Principles

Functional Architecture Model

A functional architecture model is a set of functions and their sub-functions that defines the transformations performed by the system to complete its mission.

Function and Input-Output Flow - In the context of System Architecture, functions and input-output flows are architecture entities. A function is an action that transforms inputs and generates outputs, involving data, materials, and/or energies. These inputs and outputs are the flow items exchanged between functions. The general mathematical notation of a function is $\boldsymbol{y} = f(\boldsymbol{x}, t)$, in which **y** and **x** are vectors that may be represented graphically and t = time.

In order to define the complete set of functions of the system, one must identify all the functions necessitated by the system and its derived requirements, as well as the corresponding inputs and outputs of those functions. Generally speaking, there are two kinds of functions:

1. Functions that are directly deduced from functional and interface requirements. These functions express the expected services of a system necessary to meet its system requirements.
2. Functions that are derived and issued from the alternative solutions of physical architecture model and are dependent upon the result of the design; additionally, they rely upon on technology choice to implement the logical architecture model elements.

Functional Hierarchy/Decomposition of Functions - At the highest level of a hierarchy (Figure 1), it is possible to represent a system as a unique, central function (defined as the system's mission) that in many ways is similar to a "black box" ("F0" in plan A-0 in Figure 1). In order to understand, in detail, what the system does, this "head-of-hierarchy" (F0) is broken down into sub-functions (F1, F2, F3, F4) grouped to form a sub-level of the hierarchy (plan A0), and so on. Functions of the last level of a functional hierarchy can be called leaf-functions (F21, F22, F23, F24 in plan A2). Hierarchies (or breakdowns) decompose a complex or global function into a set of functions for which physical solutions are known, feasible, or possible to imagine.

This view of functional hierarchy represents a static view of functions which would be populated at different levels over a number of iteration, depending upon the synthesis approach used. In general, it is not created by a single top-down decomposition. A static functional hierarchy on its own does not represent how effectively the flows of inputs and outputs are exchanged, and may need to be viewed alongside the other models below.

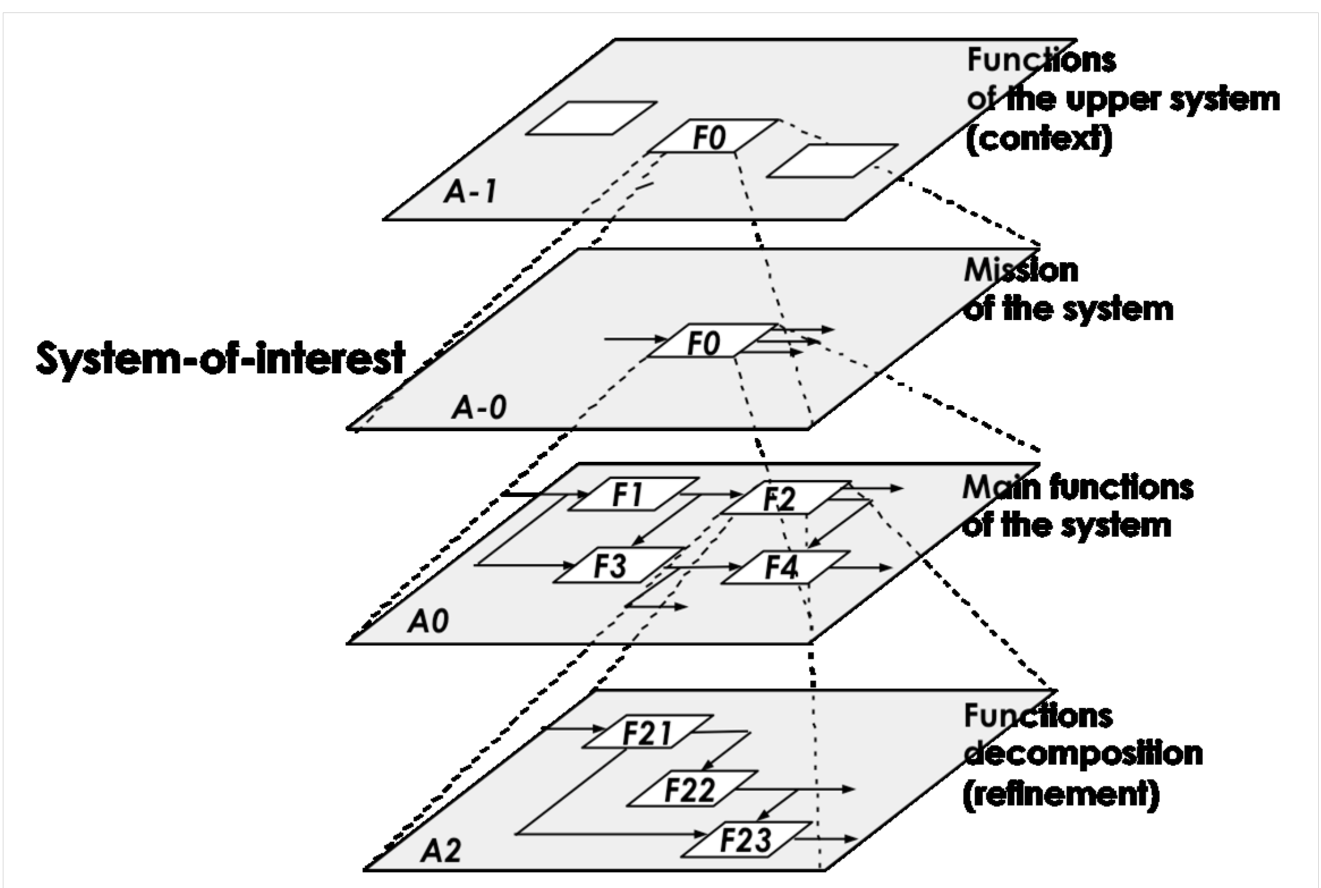


Figure 1. Decomposition of Functions (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Behavioral Architecture Model

A behavioral architecture model is an arrangement of functions and their sub-functions as well as interfaces (inputs and outputs) that defines the execution sequencing, conditions for control or data-flow, and performance level necessary to satisfy the system requirements ISO/IEC/IEEE 26702 (ISO 2007). A behavioral architecture model can be described as a set of inter-related scenarios of functions and/or operational modes.

Control (Trigger) - A control flow is an element that activates a function as a condition of its execution. The state of this element, or the condition it represents, activates or deactivates the function (or elements thereof). A control flow can be a signal or an event, such as a switch being moved to the *on* position, an alarm, a trigger, a temperature variation, or the push of a key on a keyboard.

Scenario (of Functions) - A scenario of functions is a chain of functions that are performed as a sequence and synchronized by a set of control flows to work to achieve a global transformation of inputs into outputs, as seen in the figures below. A scenario of functions expresses the dynamic of an upper level function. A behavioral architecture is developed by considering both scenarios for each level of the functional hierarchy and for each level of the system hierarchy. When representing scenarios of functions and behavioral architecture models, it is appropriate to use diagrams as modeling techniques, such as functional flow block diagrams (FFBD) (Oliver, Kelliher, and Keegan 1997) or activity diagrams, developed with SysML (OMG 2010). Figures 2 and 3 provide examples of these diagrams.

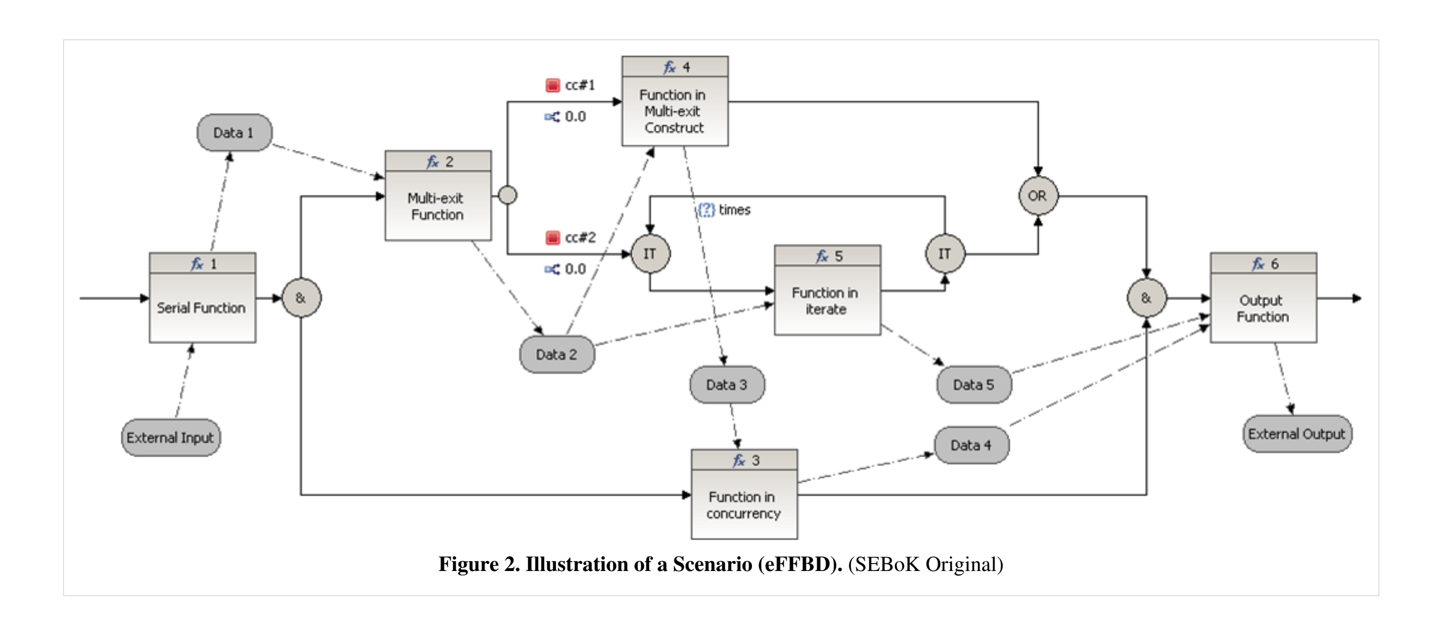


Figure 2. Illustration of a Scenario (eFFBD). (SEBoK Original)

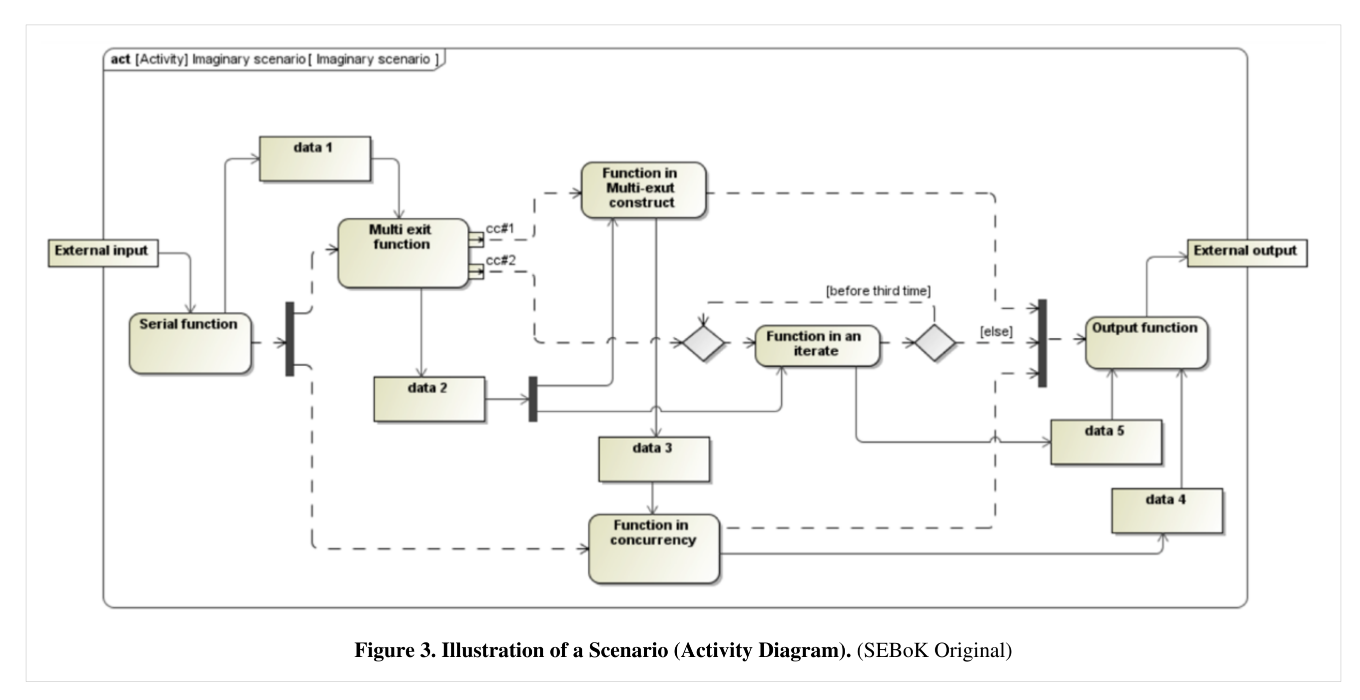


Figure 3. Illustration of a Scenario (Activity Diagram). (SEBoK Original)

Operational Mode - A scenario of functions can be viewed by abstracting the transformation of inputs into outputs of each function and focusing on the active or non-active state of the function and its controls. This view is called a *scenario of modes*, which is a chain of modes performed as a sequence of transitions between the various modes of the system. The transition from one mode to another is triggered by the arrival of a control flow (event/trigger). An action (function) can be generated within a transition between two modes following the arrival of an event or a trigger, as demonstrated in Figure 4 below.

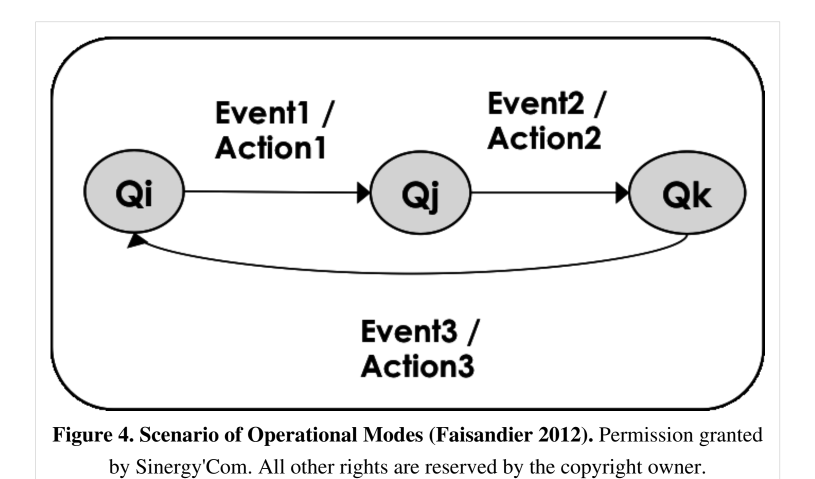


Figure 4. Scenario of Operational Modes (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Behavioral Patterns - When defining scenarios or behavioral architecture models, architects may opt to recognize and use known models to represent the expected transformations and behaviors. Patterns are generic basic models that may be more or less sophisticated depending on the complexity of the treatment. (Gamma, Helm, Johnson, and Vlissides 1995) A pattern can be represented with different notations. Behavioral patterns are classified into several categories, which can be seen in the following examples (see also SEBoK Part 2: Patterns of Systems Thinking):

- Basic patterns or constructs linking functions - such as sequence, iteration, selection, concurrence, multiple exits, loops with an exit, and replication.
- Complex patterns - such as monitoring a treatment, exchanging a message, man machine interfaces, modes monitoring, real-time monitoring of processes, queue management, and continuous monitoring with supervision.
- Failure detection, identification, and recovery (FDIR) patterns - such as passive redundancies, active redundancies, semi-active redundancies, and treatments with reduced performance.

Temporal Architecture Model

A temporal architecture model is a classification of the functions of a system that is derived according to the frequency level of execution. Temporal architecture models include the definition of synchronous and asynchronous aspects of functions. The decision monitoring that occurs inside a system follows the same temporal classification because the decisions are related to the monitoring of functions.

Temporal and Decisional Hierarchy Concept - Not every function of a system is performed at the same frequency. The frequencies change depending on the time and the manner in which the functions are started and executed. One must therefore consider several classes of performance. There are synchronous functions that are executed cyclically and asynchronous functions that are executed following the occurrence of an event or trigger.

To be more specific, *real-time* systems and *command-control* systems combine cyclical operations (synchronous) and factual aspects (asynchronous). Cyclical operations consist of sharing the execution of functions according to frequencies, which depend on either the constraints of capture or dispatching the input/output and control flows. Two types of asynchronous events can be distinguished:

1. Disturbances on High Frequencies (bottom of figure 5) - Decisions that are made at either the level they occur or one level above. The goal is to deter disturbances from affecting the low frequencies so that the system continues to achieve its mission objectives. This is the way to introduce exception operations, with the typical example relating to operations concerns, breakdowns, or failures.
2. Changes on Low Frequencies (top of figure 5) - Decisions pertaining to changes that are made at the upper levels. The ultimate goal is to transmit them toward bottom levels to implement the modifications. A typical example relates to operator actions, maintenance operations, etc.

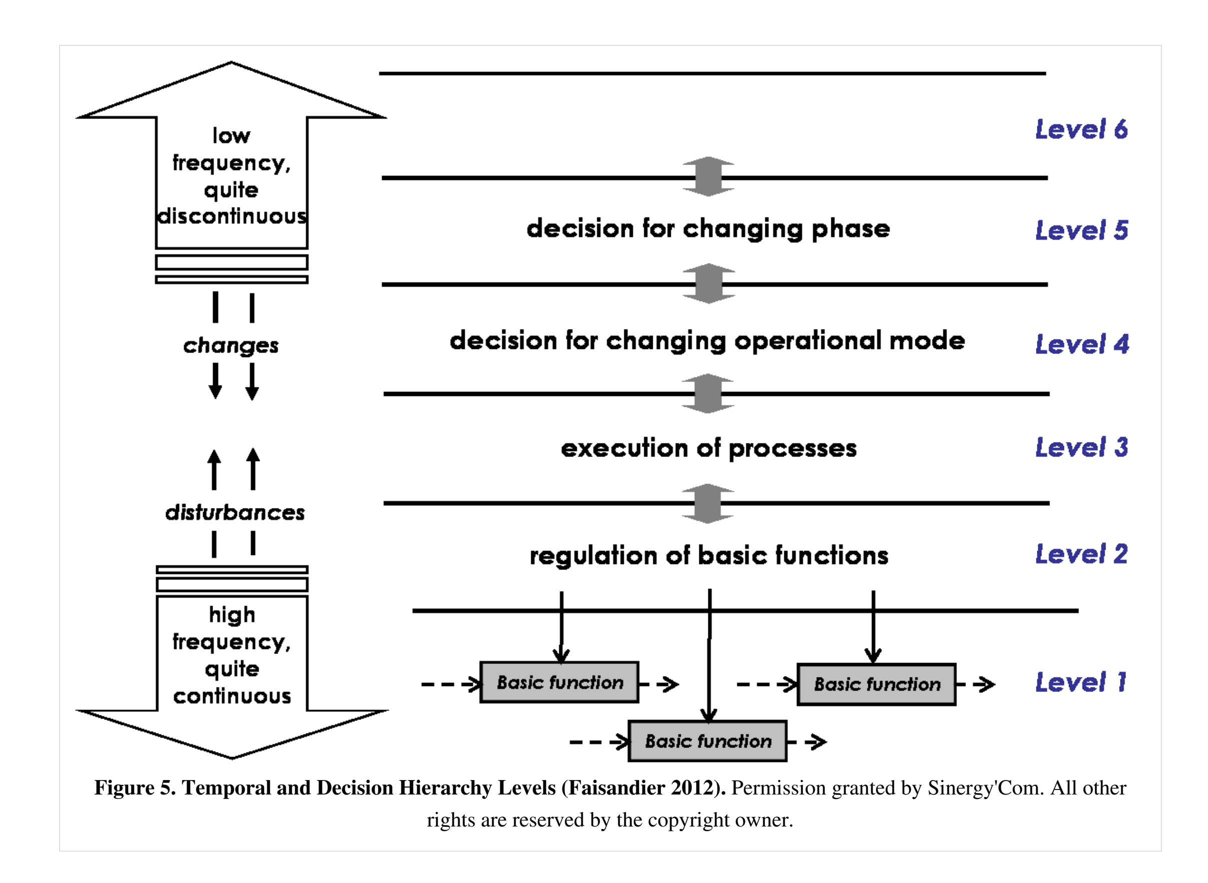


Figure 5. Temporal and Decision Hierarchy Levels (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Process Approach

Purpose

The purpose of the Logical Architecture Model Development is to define, select, and synthesize a system's logical architecture model to provide a framework against which to verify that a future system will satisfy its system requirements in all operational scenarios, within which trade-offs between system requirements can be explored in developing such systems.

Generic inputs to the process include system requirements, generic architecture patterns that architects identify and use to answer requirements, outcomes from system analysis processes, and feedback from system verification and validation processes. Depending on the Life Cycle Model that is chosen, there will be iterations through which these inputs and outputs, and the relationships between them evolve and change throughout the process (see also Applying Life Cycle Processes).

Generic outputs from the process are either a single logical architecture model or a set of candidate logical architecture models together with the selected independent logical architecture model and a rationale for its selection. They include, at minimum, views and models. These involve functional, behavioral and temporal views; a traceability matrix between logical architecture model elements and system requirements.

Activities of the Process

Major activities and tasks performed during this process include the following:

- Identify and analyze functional and behavioral elements:
  - Identify functions, input-output flows, operational modes, transition of modes, and operational scenarios from system requirements by analyzing the functional, interface, and operational requirements.
  - Define necessary inputs and controls (energy, material, and data flows) to each function and outputs that result in the deduction of the necessary functions to use, transform, move, and generate the input-output flows.
- Assign system requirements to functional and behavioral elements:
  - Formally characterize functions expressions and their attributes through the assignment of performance, effectiveness, and constraints requirements. In particular, study the temporal aspects from requirements to assign duration, response time, and frequency to functions.
  - Formally characterize the input, output, and control flows expressions and their attributes through assignment of interface, effectiveness, operational, temporal and constraints requirements.
  - Establish traceability between system requirements and these functional and behavioral elements.
- Define candidate logical architecture models For each candidate:
  - Analyze operational modes as stated in the system requirements (if any) and/or use previously defined elements to model sequences of operational modes and the transition of modes. Eventually decompose the modes into sub-modes and then establish for each operational mode one or several scenarios of functions recognizing and/or using relevant generic behavioral patterns.
  - Integrate these scenarios of functions in order to get a behavioral architecture model of the system (a complete picture of the dynamic behavior).
  - Decompose previously defined logical elements as necessary to look towards implementation.
  - Assign and incorporate temporal constraints to previously defined logical elements, such as the period of time, duration, frequency, response-time, timeout, stop conditions, etc.
  - Define several levels of execution frequency for functions that correspond to levels of decision, in order to monitor system operations, prioritize processing on this time basis, and share out functions among those execution frequency levels to get a temporal architecture model.
  - Perform functional failure modes and effects analysis and update the logical architecture elements as necessary.
  - Execute the models with simulators (when possible) and tune these models to obtain the expected characteristics.
- Synthesize the selected independent logical architecture model:
  - Select the logical architecture by assessing the candidate logical architecture models against assessment criteria (related to system requirements) and compare them, using the system analysis process to perform assessments and decision management process for the selection (see the System Analysis and Decision Management topics). This selected logical architecture model is called *independent logical architecture model* because, as much as possible, it is independent of implementation decisions.
  - Identify and define derived logical architecture model elements created for the necessity of design and corresponding with the derived system requirements. Assign these requirements to the appropriate system (current studied system or external systems).
  - Verify and validate the selected logical architecture models (using as executable models as possible), make corrections as necessary, and establish traceability between system requirements and logical architecture model elements.
- Feedback logical architecture model development and system requirements. This activity is performed after the physical architecture model development process:

- Model the *allocated logical architecture* to systems and system elements, if such a representation is possible, and add any functional, behavioral, and temporal elements as needed to synchronize functions and treatments.
- Define or consolidate derived logical and physical elements induced by the selected logical and physical architecture models. Define the corresponding derived requirements and allocate them to appropriate logical and physical architectures elements. Incorporate these derived requirements into the requirements baselines of impacted systems.

Artifacts, Methods and Modeling Techniques

Logical architecture descriptions use modeling techniques that are grouped under the following types of models. Several methods have been developed to support these types of models (some are executable models):

- Functional Models – These include models such as the structured analysis design technique (SADT/IDEF0), system analysis & real time (SA-RT), enhanced Functional Flow Block Diagrams (eFFBD), and the function analysis system technique (FAST).
- Semantic Models- These include models such as entities-relationships diagrams, class diagrams, and data flow diagrams.
- Dynamic Models – These include such models as state-transition diagrams, state-charts, eFFBDs, state machine diagrams (SysML), activity diagrams (SysML) (OMG. 2010), and petri nets.

Depending on the type of domain (e.g. defense, enterprise), architecture frameworks provide descriptions that can help to represent additional aspects/views of architectures - see the section 'Enterprise Architecture Frameworks & Methodologies' in Enterprise Systems Engineering Key Concepts. See also practical means for using general templates related to ISO/IEC/IEEE 42010 (ISO 2011).

Practical Considerations

As stated above, the purpose of the logical architecture model is to provide a description of what a system must be able to do to satisfy the stated need. This should help to ensure that the needs and/or concerns of all stakeholders are addressed by any solution, and that innovative solutions, as well as those based on current solution technologies, can be considered. In practice it is human nature for problem stakeholders to push their own agendas and for solution architects or designers to offer their familiar solutions. If a logical architecture model is not properly enforced with the chosen life cycle, it is easy for both problem and solution stakeholders to ignore it and revert to their own biases (see Part 5 Enabling Systems Engineering). This is exacerbated if the logical architecture model becomes an end in its own right or disconnected from the main lifecycle activities. This can occur either through the use of abstract language or notations, levels of detail, time taken, or an overly complex final architecture that does not match the purpose for which it was created. If the language, scope, and timeliness of the architecture are not matched to the problem stakeholder or solution providers, it is easier for them to overlook it. Key pitfalls and good practices which can help to avoid problems related to logical architecture model are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in developing logical architecture are provided in Table 1.

Table 1. Pitfalls with Logical Architecture Development. (SEBoK Original)

Pitfall	Description
Problem Relevance	The logical architecture model should relate back to the operational scenarios produced by mission analysis.
Inputs for Architecture Model	The major input for architecture definition activity involves the set of system requirements and the instances in which they do not address the right level of architecture. The consequence is that the architect allows the requirements to fall to the side and invents a solution with what he or she understands through the input.
Decomposition Too Deep	A common mistake made by many beginners in architecture consists of decomposing the functions too deeply or having too many functions and input/output flows in scenarios or in the functional architecture model of the current system block.
Not Considering Inputs and Outputs Together with Functions	A common mistake is to consider only the actions supported by functions and decomposing them, while forgetting the inputs and the outputs or considering them too late. Inputs and outputs are integral parts of a function.
Considering Static Decomposition of Functions Only	Static function decomposition is the smallest functional architecture model task and answers the basic question, "How is this done?" The purpose of the static decomposition is to facilitate the management or navigation through the list of functions. The static decomposition should be established only when scenarios have been created and the logical architecture is close to complete.
Mixing Governance, Management, and Operation	Governance (strategic monitoring), management (tactical monitoring), and basic operations are often mixed in complex systems. Logical architecture model should deal with behavioral architecture model as well as with temporal architecture model.

Proven Practices

Some proven practices gathered from the references are provided in Table 2.

Table 2. Proven Practices with Logical Architecture Development. (SEBoK Original)

Practice	Description
Constitute Scenarios of Functions	Before constituting a decomposition tree of functions, one must model the behavior of the system, establish scenarios of functions, and decompose functions as scenarios of sub-functions.
Analysis and Synthesis Cycles	When facing a system that contains a large number of functions, one should attempt to synthesize functions into higher abstraction levels of functions with the assistance of criteria. Do not perform analysis only; instead, conduct small cycles of analysis (decomposition) and synthesis. The technique of using scenarios includes this design practice.
Alternate Functional and Behavioral Views	A function (action verb; e.g. "to move") and its state of execution/operational mode (e.g. "moving") are two similar and complimentary views. Utilize this to consider a behavioral view of the system that allows for the transition from one operational mode to another.
The Order to Create a Scenario Of Functions	When creating a scenario of functions, it is more efficient to first establish the (control) flow of functions, then to add input and output flows, and finally to add triggers or signals for synchronization.

References

Works Cited

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA, USA: Addison-Wesley.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

ISO/IEC. 2007.*Systems Engineering – Application and Management of the Systems Engineering Process.*Geneva, Switzerland: International Organization for Standards (ISO)/International Electronical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering complex systems with models and objects*. New York, NY, USA: McGraw-Hill.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2, July 2010. http://www.omg.org/technology/documents/spec_catalog.htm.

Primary References

ANSI/IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

INCOSE. 2015. *Systems Engineering Handbook* - A Guide for System Life Cycle Processes and Activities", version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC. 2007. *Systems Engineering – Application and Management of the Systems Engineering Process.* Geneva, Switzerland: International Organization for Standards (ISO)/International Electronical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Maier, M. and E. Rechtin. 2009. *The Art of Systems Architecting,* 3rd ed. Boca Raton, FL, USA: CRC Press.

Additional References

Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. 1977. *A Pattern Language: Towns, Buildings, Construction*. New York, NY, USA: Oxford University Press.

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering Complex Systems with Models and Objects*. New York, NY, USA: McGraw-Hill.

The Open Group. 2011. *TOGAF*, version 9.1. Hogeweg, The Netherlands: Van Haren Publishing. Accessed August 29, 2012. Available at: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g116.

Zachman, J. 2008. "John Zachman's Concise Definition of The Zachman Framework™." Zachman International Enterprise Architecture. Accessed August 29, 2012. Available at: http:/ / www. zachman. com/ about-the-zachman-framework.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Physical Architecture Model Development

Physical Architecture Model Development may be used as a task of the activity "Develop candidate architectures models and views", or a sub-process of the System Architecture Definition process (see **System Architecture** article).Its purpose is to elaborate models and views of a physical, concrete solution that accommodates the logical architecture model and satisfies and trades-off system requirements. Once a logical architecture model is defined (see Logical Architecture Model Development), concrete physical elements have to be identified that can support functional, behavioral, and temporal features as well as the expected properties of the system deduced from non-functional system requirements (e.g. constraint of replacement of obsolescence, and/or continued product support).

A physical architecture model is an arrangement of physical elements, (system elements and physical interfaces) that provides the solution for a product, service, or enterprise. It is intended to satisfy logical architecture elements and system requirements ISO/IEC/IEEE 26702 (ISO 2007). It is implementable through technological system elements. System requirements are allocated to both the logical and physical architectures. The resulting system architecture is assessed with system analysis and when completed becomes the basis for system realization.

In some cases, particularly when multiple systems are to be defined to a common physical architecture model, one of the drivers for the physical architecture model may be interface standards; these physical interfaces may well be one of the most important concerns for these systems. It is quite possible that such interface standards are mandated at a high level in the system requirements. On the other hand, it is equally possible for standards to be derived during physical architecture model development and these can be critical enablers for desirable engineering outcomes, such as: families of systems, technology insertion, interoperability and “open systems”. For example, today’s video, hi-fi, and computer systems have all benefited from adoption of interface standards. Other examples exist in most fields of engineering from nuts and bolts, plumbing, electrical installations, rail gauges, TCP/IP, IT systems and software to modular defense and space systems.

Note: The term *Physical Architecture* is a contraction of the expression *Physical View of the System Architecture*.

Concepts and Principles

System Element, Physical Interface, and Physical Architecture Model

A system element is a discrete part of a system that can be implemented to fulfill design properties. A system element can be hardware, software, data, humans, processes (e.g., processes that provide a service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities (e.g., water, organisms, and minerals), or any combination of these ISO/IEC/IEEE 15288 (ISO 2015). A physical interface binds two system elements together; this is similar to a link or a connector. Table 1 provides some examples of system elements and physical interfaces.

Table 1. Types of System Elements and Physical Interfaces. (SEBoK Original)

Element	Product System	Service System	Enterprise System
System Element	• Hardware Parts (mechanics, electronics, electrical, plastic, chemical, etc.) • Operator Roles • Software Pieces	• Processes, Data Bases, Procedures, etc. • Operator Roles • Software Applications	• Corporate, Direction, Division, Department, Project, Technical Team, Leader, etc. • IT Components
Physical Interface	* Hardware Parts, Protocols, Procedures, etc.	* Protocols, Documents, etc.	* Protocols, Procedures, Documents, etc.

A complex system composed of thousands of physical and/or intangible parts may be structured in several layers of systems and system elements. The number of elements in a level of the structure of one system is limited to only a few, in order to facilitate managing the system definition; a common guideline is *five plus or minus two* elements (see illustration in Figure 1).

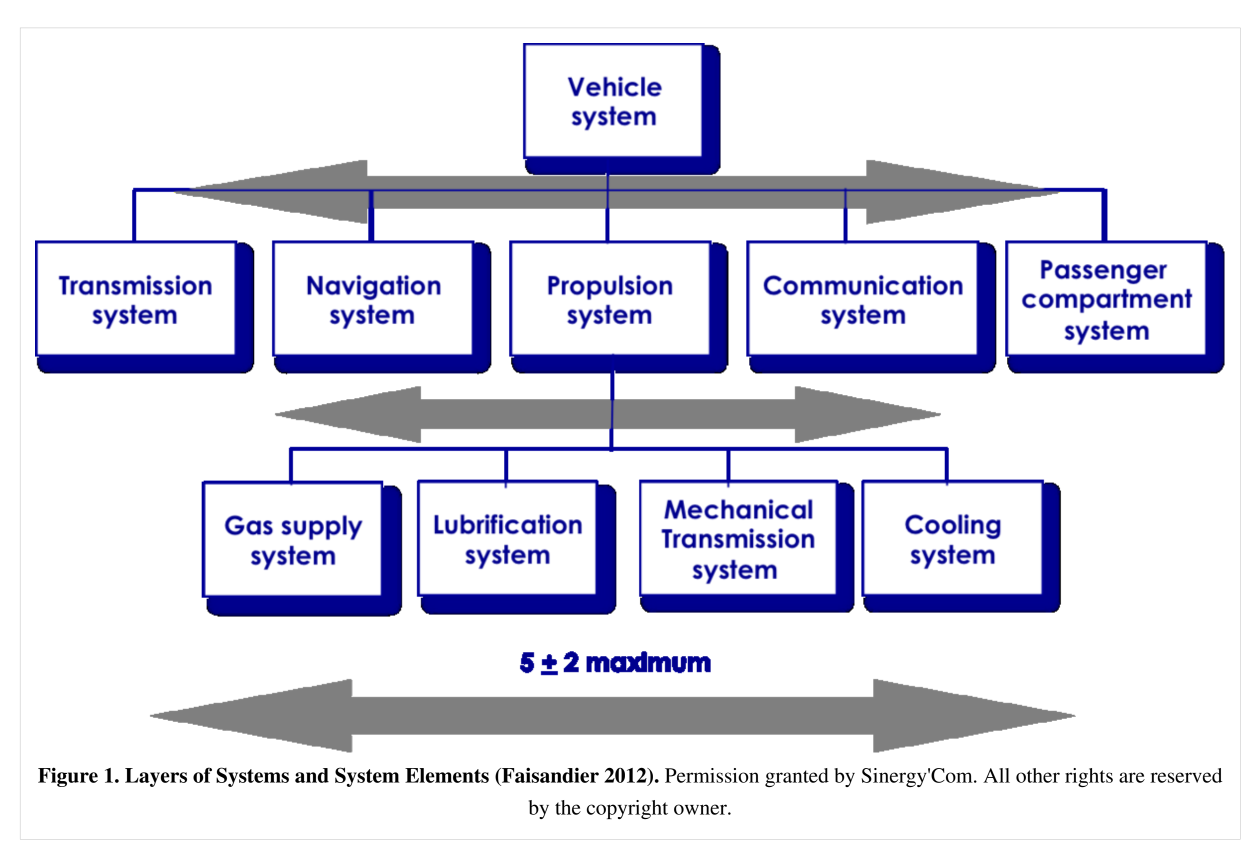


Figure 1. Layers of Systems and System Elements (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

A physical architecture model is built from systems, system elements, and all necessary physical interfaces between these elements, as well as from external elements (neighboring or enabling systems and/or system elements in the considered layer and concerned elements in the context of the global system-of-interest) - see illustration in Figure 2.

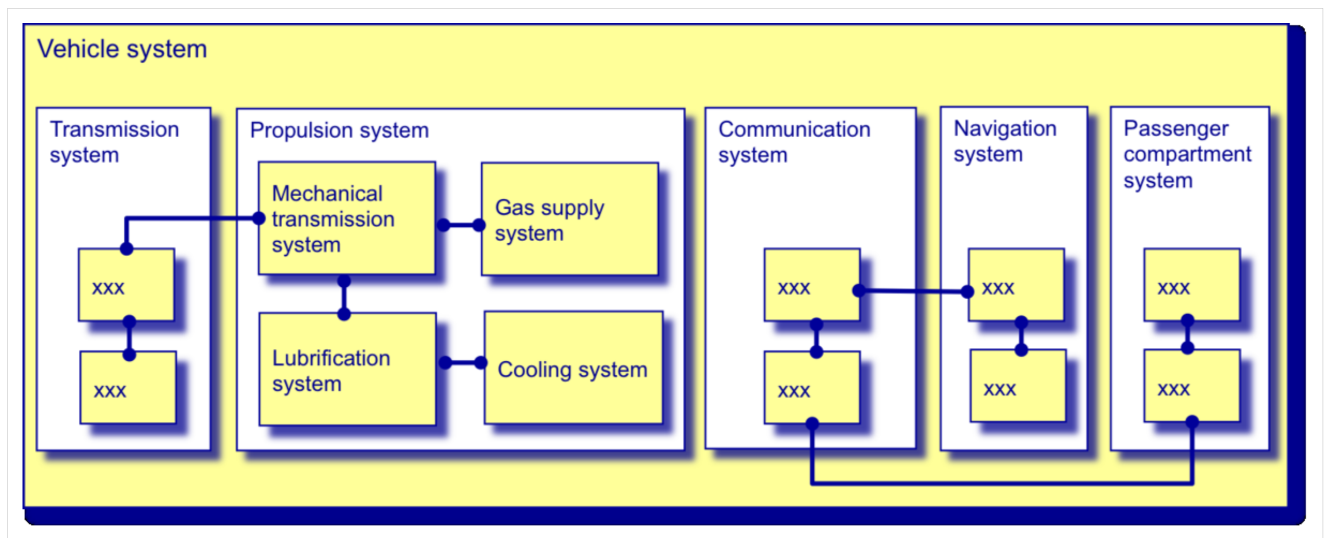


Figure 2. Physical Architecture Model Representation (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Design Property

A design property is a property that is obtained during system architecture and created through the assignment of non-functional requirements, estimates, analyses, calculations, simulations of a specific aspect, or through the definition of an existing element associated with a system element, a physical interface, and/or a physical architecture. If the defined element complies with a requirement, the design property will relate to (or may equal) the requirement. Otherwise, one has to identify any discrepancy that could modify the requirement or design property, and detect any deviations.

Stakeholders have concerns that correspond to the expected behavior of a system within operational, environmental, and/or physical constraints as well as to more general life cycle constraints. Stakeholder requirements and system requirements express these concerns as expected capabilities from the system (e.g., usability, interoperability, security, expandability, environment suitability, etc.). Architects and/or designers identify these capabilities from requirements and deduce corresponding quantitative or qualitative design properties to properly equip their physical architecture model (e.g., reliability, availability, maintainability, modularity, robustness, operability, climatic environment resistance, dimensions limits, etc.). For further discussion on how some of these properties may be included in architecture and design, please see the article Systems Engineering and Specialty Engineering in the Related Disciplines knowledge area (KA).

Allocation of Logical Elements to Physical Elements and Partitioning

Developing a candidate physical architecture model for a system consists of first identifying the system elements that can perform functions of the logical architecture model as well as identifying the interfaces capable of carrying out the input-output flows and control flows. When identifying potential elements, a systems engineer needs to allocate design properties within the logical architecture; these properties are deduced from the system requirements. Partitioning and allocation are activities to decompose, gather, or separate functions in order to facilitate the identification of feasible system elements that support these functions. Either they exist and can be reused or re-purposed, or they can be developed and technically implemented.

Partitioning and allocation use criteria to find potential affinities between functions. Systems engineers use system requirements and/or design properties as criteria to assess and select candidate system elements and partitions of functions, such as similar transformations within the same technology, similar levels of efficiency, exchange of the same type of input-output flows (information, energy, and materials), centralized or distributed controls, execution

with close frequency level, dependability conditions, environment resistance level, and other enterprise constraints.

A concurrent engineering approach is necessary when several different sets of technologies, knowledge, and skills are necessary to establish a candidate physical architecture model. This is particularly true during the partition and allocation of functions to various system elements, in which the systems engineer must account for compatibility issues and emergent properties. (see SEBoK Part 2: Synthesizing Possible Solutions for a discussion of possible approaches)

Developing Candidate Physical Architecture Models

The goal of physical architecture model development activities is to provide the best possible physical architecture model made of suitable systems, technological system elements, and physical interfaces (i.e., the architecture that answers, at best, all system requirements, depending on agreed limits or margins of each requirement). The best way to do this is to produce several candidate physical architecture models, assess and compare them, and then select the most suitable one.

A candidate physical architecture model is elaborated according to affinity criteria in order to build a set of system elements (i.e., separate, gather, connect, and disconnect the network of system elements and their physical interfaces). These criteria are the same as those used for partitioning and allocating functions to system elements. The physical architecture model development may be focused in different ways, for example, it may address:

- Reduction in the number of physical interfaces
- System elements that can be tested separately
- Compatible technology
- Measures of the proximity of elements in space
- Ease of handling (weight, volume, and transportation facilities)
- Optimization of resources shared between elements
- Modularity (i.e. elements have low interdependence)
- Resilience (i.e. elements which are highly reliable, maintainable or replaceable)

Evaluating and Selecting the Preferred Candidate

Viable physical architecture models enable all required functions or capabilities specified in the logical architecture model to be realized. Architecture and design activity includes evaluation to obtain a balance among design properties, costs, risks, etc. Generally, the physical architecture model of a system is determined more strongly by non-functional requirements (e.g., performance, safety, security, environmental conditions, constraints, etc.) than by functions. There may be many (physical) ways to establish functions but fewer ways of satisfying non-functional requirements. The preferred physical architecture model represents the selection of system elements, their physical relationships, and interfaces. Typically this physical architecture will still leave further systems engineering to be undertaken to achieve a fully optimized system after any remaining trade-offs are made and algorithms and parameters of the system are finalized.

Certain analyses (efficiency, dependability, cost, risks, etc.) are required to get sufficient data that characterize the global behavior and structure of the candidate architectures in regard to system requirements; this is often broadly referred to as system analysis. Other analyses and assessments require knowledge and skills from the different involved technologies and specialities (mechanics, electronics, software, thermodynamics, electro-magnetic compatibility, safety, security etc.). They are performed through corresponding specialist analysis of the system.

Legacy Systems and Systems of Systems

Few systems come into existence or operate without interacting with others in a system context. These interactions may be with other operational systems, or maintenance and support systems, which in turn may be legacy (already in use) or future legacy (under development and likely to operate with the system of interest in the future).

The best chosen approach will be dependent on the strength of interactions between the System-of-Interest (glossary) (SoI) and its wider context. While these interactions are small, they may be accounted for by defining a set of static external interface requirements (for example technical standards) with which the system must comply, by including these as constraints in the system requirements and ensuring compliance through design assurance.

Where the interactions are more intense, for example where continuous information is to be exchanged with other systems, these will have to be recognized as part of a system of systems context and will instead be considered as part of an enterprise systems engineering approach.

Another important consideration may be the sharing of technology or system elements between the SoI and other systems, often as part of a family of systems (many examples occur in automotive and aerospace industries) or the re-use of system elements from existing legacy. Here a degree of top-down or middle-out design work will be necessary to ensure the system of interest embodies the required system elements, while conforming as far as possible to the stakeholder and system requirements, with any compromises being understood and managed.

If a System-of-Interest is intended to be used in one or more service systems or system of systems configurations this will affect its physical architecture model. One of the features of these SoS is the late binding of component systems in use. Such component systems must be architected with open or configurable interfaces, must have clearly defined functions packaged in such a way as to be relevant to the SoS using them, and must include some method by which they can be identified and included in the SoS when needed.

Both service systems and SoS will be defined by a high level physical architecture model, which will be utilized to define the relevant SoS relationships, interfaces, and constraints that should be included in Concept Definition. The results will be embedded in the stakeholder and system requirements and handled through interface agreements and across-project communication during development, realization, and use.

See SEBoK Part 4 Applications of Systems Engineering for more information on special considerations for architecting SoS.

Process Approach

Purpose

The purpose of the Physical Architecture Model Development is to define, select, and synthesize a system physical architecture model which can support the logical architecture model. A physical architecture model will have specific properties to address stakeholder concerns or environmental issues and to satisfy system requirements.

Because of the evolution of the context of use or technological possibilities, the physical architecture which is composed of system elements is supposed to evolve along the life cycle of the system in order for it to continue to perform its mission within the limits of its required effectiveness. Depending on whether or not evolution impacts logical architecture model elements, allocations to system elements may change. A physical architecture model is equipped with specific design properties (glossary) to continuously challenge the evolution.

Generic inputs include the selected logical architecture model, system requirements, generic patterns and properties that architects identify and utilize to answer requirements, outcomes from system analysis, and feedback from system verification and system validation.

Generic outputs are the selected physical architecture model, allocation matrix of functional elements to physical elements, traceability matrix with system requirements, stakeholder requirements of each system and system element composing the physical architecture model, and rejected solutions.

Activities of the Process

Major activities and tasks to be performed during this process include the following:

- Partition and allocate functional elements to system elements:
  - Search for system elements or technologies able to perform functions and physical interfaces to carry input-output and control flows. Ensure system elements exist or can be engineered. Assess each potential system element using criteria deduced from design properties (themselves deduced from non-functional system requirements).
  - Partition functional elements (functions, scenarios, input-outputs, triggers, etc.) using the given criteria and allocate partitioned sets to system elements (using the same criteria).
  - When it is impossible to identify a system element that corresponds to a partitioned functional set, decompose the function until the identification of implementable system elements is possible.
  - Check the compatibility of technologies and the compatibility of interfaces between selected system elements.
- Constitute candidate physical architecture models.
  - Because partitioned sets of functions can be numerous, there are generally too many system elements. For defining controllable architectures, system elements have to be grouped into higher-level system elements known as system element groups, often called sub-systems in industry.
  - Constitute several different system element groups corresponding to different combinations of elementary system elements. One set of system element groups plus one or several non-decomposable system elements form a candidate physical architecture model of the considered system.
  - Represent (using patterns) the physical architecture model of each system element group connecting its system elements with physical Interfaces that carry input-output flows and triggers. Add physical interfaces as needed; in particular, add interfaces with external elements to the system element group.
  - Represent the synthesized physical architecture of the considered system built from system element groups, non-decomposable system, and physical interfaces inherited from the physical architecture model of system element groups.
  - Enhance the physical architecture model with design properties such as modularity, evolution capability, adaptability to different environments, robustness, scalability, resistance to environmental conditions, etc.
  - If possible, use executable architecture prototypes (e.g., hardware-software (HW-SW)-in-the-loop prototypes) for identifying potential deficiencies and correct the architecture as needed.
- Assess physical architecture model candidates and select the most suitable one:
  - Use the system analysis process to perform assessments (see the System Analysis topic).
  - Use the Decision Management process to support the trades and selection of the preferred alternative (see the Decision Management topic).
- Synthesize the selected physical architecture model:
  - Formalize physical elements and properties. Verify that system requirements are satisfied and that the solution is realistic.
  - Identify the derived physical and functional elements created for the necessity of architecture and design and the corresponding system requirements.
  - Establish traceability between system requirements and physical elements as well as allocate matrices between functional and physical elements.

Artifacts, Methods and Modeling Techniques

Physical architecture descriptions use modeling techniques to create and represent physical architectures. Some common physical models include structural blocks, mass, layout and other models. Modeling techniques may be:

- Physical block diagrams (PBD)
- SysML block definition diagrams (BDD)
- Internal block diagrams (IBD) (OMG 2010)
- Executable architecture prototyping
- Etc.

Depending on the type of domain for which it is to be used (defense, enterprise, etc.), architecture frameworks may provide descriptions that can help to trade-off candidate architectures. Please see section 'Enterprise Architecture Frameworks & Methodologies' in Enterprise Systems Engineering Key Concepts.

Practical Considerations

Key pitfalls and good practices related to physical architecture development are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in performing physical architecture model development are provided in Table 3.

Table 3. Pitfalls with Physical Architecture Development. (SEBoK Original)

Pitfall	Description
Too Many Levels in a Single System Block	The current system block includes too many levels of decomposition. The right practice is that the physical architecture model of a system block is composed of one single level of systems and/or system elements.
No Logical Architecture Model	The developers perform a direct passage from system requirements to physical architecture model without establishing a logical architecture model; this is a common wrong practice that mainly takes place when dealing with repeating systems and products because the functions are already known. The issue is that a function is always associated with input-output flows defined in a specific domain set. If the domain set changes, the performance of the function can become invalid.
Direct Allocation on Technologies	At a high level of abstraction of multidisciplinary systems, directly allocating the functions onto technologies of the lowest level of abstraction, such as hardware or software, does not reflect a system comprehension. The right practice is to consider criteria to decompose the architecture into the appropriate number of levels, alternating logical and physical before reaching the technology level (the last level of the system).

Proven Practices

Some proven practices gathered from the references are provided in Table 4.

Table 4. Proven Practices with Physical Architecture Development. (SEBoK Original)

Practice	Description
Modularity	Restrict the number of interactions between the system elements and consider the modularity principle (maximum of consistency inside the system element, minimum of physical interfaces with outside) as the right way for architecting systems.
Focus on Interfaces	Focusing on interfaces rather than on system elements is another key element of a successful architecture and design for abstract levels of systems.

References

Works Cited

ISO/IEC. 2007. Systems Engineering – Application and Management of The Systems Engineering Process. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2, July 2010. http://www.omg.org/technology/documents/spec_catalog.htm.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Primary References

ANSI/IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

INCOSE. 2015. *Systems Engineering Handbook* - A Guide for System Life Cycle Processes and Activities", version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Additional References

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting,* 3rd ed. Boca Raton, FL, USA: CRC Press.

Holland, J.H. 2006. "Studying Complex Adaptive Systems." *Journal of Systems Science and Complexity.*19(1): 1-8. http://hdl.handle.net/2027.42/41486

Thome, B. 1993. *Systems Engineering, Principles & Practice of Computer-Based Systems Engineering.* New York, NY, USA: Wiley.

The Open Group. 2011. *TOGAF*, version 9.1. Hogeweg, The Netherlands: Van Haren Publishing. Accessed August 29, 2012. Available at: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g116.

Zachman, J. 2008. "John Zachman's Concise Definition of The Zachman Framework™." Zachman International Enterprise Architecture. Accessed August 29, 2012. Available at: http:/ / www. zachman. com/ about-the-zachman-framework.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Design

The purpose of the System Design is to supplement the system architecture providing information and data useful and necessary for implementation of the system elements. Design definition is the process of developing, expressing, documenting, and communicating the realization of the architecture of the system through a complete set of design characteristics described in a form suitable for implementation.

Concepts and principles

Design Notion

In industrial practices, the term *design* is often used to mean both Architecture (glossary) and Design (glossary). In the recent past, professionals used the term *design* when they dealt with simpler technological products - ones that do not include several different and interconnected technological components such as hardware, software, operators, services, etc. In the development of new multi-technology products and services, professionals have recognized the usefulness of the notion of *system* in dealing with complexity (glossary) (interconnections level, multi-techno, emergence, etc.).

It was due to complexity that structuring the elements that comprise a system became necessary. This structure explains the functional, behavioral, temporal, physical, and other aspects of a system as described in System Architecture. Practitioners found the term *structure* inadequate to describe all of these aspects of a system. The terms *architecture* and *architectural design* have been used for approximately 30 years, especially in software intensive systems and other domains, such as the space industry. The set of different types and interrelated structures can be understood as the architecture of the system.

The trend today is to consider system architecture and system design as different and separate sets of activities, but concurrent and strongly intertwined.

System design includes activities to conceive a set of system elements that answers a specific, intended purpose, using principles and concepts; it includes assessments and decisions to select system elements that compose the system, fit the architecture of the system, and comply with traded-off system requirements. It is the complete set of detailed models, properties, and/or characteristics described into a form suitable for implementation.

Design characteristics and design enablers

Every technological domain or discipline owns its peculiar laws, rules, theories, and enablers concerning transformational, structural, behavioral, and temporal properties of its composing parts of materials, energy, or information. These specific parts and/or their compositions are described with typical design characteristics and enablers. These allow achieving the implementation of every system element through various transformations and exchanges required by design characteristics (e.g., operability level, reliability rate, speed, safeguard level) that have been assigned during the system architecture definition process.

The design definition provides the description of the design characteristics and design enablers necessary for implementation. Design characteristics include dimensions, shapes, materials, and data processing structures. Design

enablers include formal expressions or equations, drawings, diagrams, tables of metrics with their values and margins, patterns, algorithms, and heuristics.

- Examples of generic design characteristics in mechanics of solids: shape, geometrical pattern, dimension, volume, surface, curves, resistance to forces, distribution of forces, weight, velocity of motion, temporal persistence
- Examples of generic design characteristics in software: distribution of processing, data structures, data persistence, procedural abstraction, data abstraction, control abstraction, encapsulation, creational patterns (e.g., builder, factory, prototype, singleton), and structural patterns (e.g., adapter, bridge, composite, decorator, proxy)

Relation with System Architecture

System design is intended to be the link between the system architecture (at whatever point this milestone is defined in the specific application of the systems engineering process) and the implementation of technological system elements that compose the physical architecture model of the system.

Design definition is driven by specified requirements, the system architecture, and more detailed analysis of performance and feasibility. It addresses the implementation technologies and their assimilation. Design provides the "how" or "implement-to" level of the definition.

Design concerns every system element composed of implementation technologies, such as for example mechanics, electronics, software, chemistry, human operations and services for which specific engineering processes are needed. System design provides feedback to the parent system architecture to consolidate or confirm the allocation and partitioning of architectural characteristics and design properties to system elements.

Design Descriptor

A design descriptor is the set of generic design characteristics and of their possible values. If similar, but not exact system elements exist, it is possible to analyze these in order to identify their basic characteristics. Variations of the possible values of each characteristic determine potential candidate system elements.

Holistic Design

Holistic design is an approach that considers the system being designed as an interconnected whole, which is also part of something larger. Holistic concepts can be applied to the system as a whole along with the system in its context (e.g., the enterprise or mission in which the system participates), as well as the design of mechanical devices, the layout of spaces, and so forth. This approach often incorporates concerns about the environment, considering how the design will impact the environment and attempting to reduce environmental impact. Holistic design is about more than merely trying to meet the system requirements.

Process Approach

Purpose

The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture. ISO/IEC/IEEE 15288 (ISO 2015)

Generic inputs include architecture description of the parent system, system element requirements.

Generic outputs are the description of the design characteristics and design enablers necessary for implementation.

Activities of the Process

Major activities and tasks to be performed during this process include the following:

1. Initialize design definition

- Plan for technology management for the whole system. Identify the technologies (mechanics, electricity, electronics, software, biology, operators, etc.) that would compose and implement the system elements and their physical interfaces.
- Determine which technologies and system elements have a risk to become obsolete, or evolve during the operation stage of the system. Plan for their potential replacement.
- Identify types of design characteristics or properties for each technology of each system element.
- Periodically assess design characteristics and adjust as the system evolves.
- Document the design definition strategy, including the need for and requirements of any enabling systems, products, or services to perform the design.

2. Establish design characteristics and design enablers related to each system element

- Perform or consolidate or detail system requirements allocation to system elements for all requirements and system elements not fully addressed in the System Architecture process (normally, every system requirement would have been transformed into architectural entities and architectural characteristics within the System Architecture process, which are then allocated to system elements through direct assignment or some partitioning).
- Define the design characteristics relating to the architectural characteristics and check that they are implementable. Use design enablers, such as models (physical and analytical), design heuristics, etc. If the design characteristics are not feasible, then assess other design alternatives or implementation option, or perform trades of other system elements definition.
- Define the interfaces that were not defined by the System Architecture process or that need to be refined as the design details evolve. This includes both internal interfaces between the system elements and the external interfaces with other systems.
- Record the design characteristics of each system element within the applicable artifacts (they depend on the design methods and techniques used).
- Provide rationale about selection of major implementation options and enablers.

3. Assess alternatives for obtaining system elements

- Identify existing implemented system elements (COTS/NDI, reused, or other non-developed system elements). Alternatives for new system elements to be developed may be studied.
- Assess design options for the system element, using selection criteria that are derived from the design characteristics.
- Select the most appropriate alternatives.
- If the decision is made to develop the system element, rest of the design definition process and the implementation process are used. If the decision is to buy or reuse a system element, the acquisition process may be used to obtain the system element.

4. Manage the design

- Capture and maintain the rationale for all selections among alternatives and decisions for the design, architecture characteristics, design enablers, and sources of system elements.
- Assess and control the evolution of the design characteristics, including the alignment with the architecture.
- Establish and maintain traceability between design characteristics and architectural characteristics, and with requirements as necessary.
- Provide baseline information for configuration management.
- Maintain the design baseline and the design definition strategy.

Practical Considerations

Key pitfalls and proven practices related to system design are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in performing system design are provided in Table 1.

Pitfall	Description
Consider only separately the design of each system element	This would conduct to use heterogeneous implementation of a given technology or between technologies within the system-of-interest. The design strategy for the complete system is defined to search synergies and/or commonalities that could help operation and maintenance of system elements.

Proven Practices

Some proven practices gathered from the references are provided in Table 2.

Practice	Description
Architecture and design mutual support	Discipline engineers perform the design definition of each system element; they provide strong support (knowledge and competencies) to systems engineers, or architects, in the evaluation and selection of candidate system architectures and system elements. Inversely, systems engineers, or architects, must provide feedback to discipline engineers to improve knowledge and know-how.

References

Works Cited

INCOSE. 2015. *INCOSE Systems Engineering Handbook,* Version 4. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Primary References

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Additional References

Baldwin, C.Y. and K.B. Clark. 2000. *Design Rules*. Cambridge, Mass: MIT Press.

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

DoD. 2010. *DOD Architecture Framework.*Version 2.02. Arlington, VA, USA: US Department of Defense. Available at: http://cio-nii.defense.gov/sites/dodaf20/

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Analysis

System analysis allows developers to objectively carry out quantitative assessments of systems in order to select and/or update the most efficient system architecture and to generate derived engineering data. During engineering, assessments should be performed every time technical choices or decisions are made to determine compliance with system requirements.

System analysis provides a rigorous approach to technical decision-making. It is used to perform trade-off studies, and includes modeling and simulation, cost analysis, technical risks analysis, and effectiveness analysis.

Principles Governing System Analysis

One of the major tasks of a systems engineer is to evaluate the engineering data and artifacts created during the systems engineering (SE) process. The evaluations are at the center of system analysis, providing means and techniques

- to define assessment criteria based on system requirements;
- to assess design properties of each candidate solution in comparison to these criteria;
- to score globally the candidate solutions and to justify the scores; and
- to decide on the appropriate solution(s).

The Analysis and Selection between Alternative Solutions article in the Systems Approach Applied to Engineered Systems knowledge area (KA) of Part 2 describes activities related to selecting between possible system solutions to an identified problem or opportunity. The following general principles of systems analysis are defined:

- Systems analysis is based on assessment criteria based upon a problem or opportunity system description.
  - These criteria will be based around an ideal system description, which assumes a hard system problem context can be defined.
  - Criteria must consider required system behavior and properties of the complete solution, in all possible wider system contexts and environments.
  - These must consider non-functional issues such as system safety, security, etc. (Please see Systems Engineering and Specialty Engineering for additional discussion on incorporating non-functional elements.)

  - This "ideal" system description may be supported by soft system descriptions, from which additional “soft” criteria may be defined. For example, a stakeholder preference for or against certain kinds of solutions, relevant social, political or cultural conventions to be considered, etc.
- The assessment criteria should include, at a minimum, the constraints on cost and time scales acceptable to stakeholders.
- Trade studies provide a mechanism for conducting analysis of alternative solutions.
  - A trade study should consider a set of assessment criteria, with appropriate awareness of the limitations and dependencies between individual criteria.
  - Trade studies need to deal with both objective and subjective criteria. Care must be taken to assess the sensitivity of the overall assessment to particular criteria.

Trade-off studies

In the context of the definition of a system, a trade-off study consists of comparing the characteristics of each system element and of each candidate system architecture to determine the solution that best globally balances the assessment criteria. The various characteristics analyzed are gathered in cost analysis, technical risks analysis, and effectiveness analysis (NASA 2007).

Guidance on the conduct of trade studies for all types of system context are characterized in the above principles and described in more details in the Analysis and Selection between Alternative Solutions topic. Of particular interest to SE analysis are technical effectiveness, cost, and technical risk analysis.

Effectiveness Analysis

The effectiveness of an engineered system solution includes several essential characteristics that are generally gathered in the following list of analyses, including (but not limited to): performance, usability, dependability, manufacturing, maintenance or support, environment, etc. These analyses highlight candidate solutions under various aspects.

It is essential to establish a classification that limits the number of analyses to the really significant aspects, such as key performance parameters. The main difficulties of effectiveness analysis are to sort and select the right set of effectiveness aspects; for example, if the product is made for a single use, maintainability will not be a relevant criterion.

Cost Analysis

A cost analysis considers the full life cycle costs. A cost baseline can be adapted according to the project and the system. The global life cycle cost (LCC), or total ownership cost (TOC), may include examplar labor and non-labor cost items such as those indicated in Table 1.

Table 1. Types of Costs. (SEBoK Original)

Type of Cost	Description and Examples
Development	Engineering, development tools (equipment and software), project management, test-benches, mock-ups and prototypes, training, etc.
Product manufacturing or service realization	Raw materials and supplying, spare parts and stock assets, necessary resources to operation (water, electricity power, etc.), risks and nuances, evacuation, treatment and storage of waste or rejections produced, expenses of structure (taxes, management, purchase, documentation, quality, cleaning, regulation, controls, etc.), packing and storage, documentation required.
Sales and after-sales	Expenses of structure (subsidiaries, stores, workshops, distribution, information acquisition, etc.), complaints and guarantees, etc.

Customer utilization	Taxes, installation (customer), resources necessary to the operation of the product (water, fuel, lubricants, etc.), financial risks and nuisances, etc.
Supply chain	Transportation and delivery
Maintenance	Field services, preventive maintenance, regulation controls, spare parts and stocks, cost of guarantee, etc.
Disposal	Collection, dismantling, transportation, treatment, waste recycling, etc.

Methods for determining cost are described in the Planning topic.

Technical Risks Analysis

Every risk analysis concerning every domain is based on three factors:

1. Analysis of potential threats or undesired events and their probability of occurrence.
2. Analysis of the consequences of these threats or undesired events and their classification on a scale of gravity.
3. Mitigation to reduce the probabilities of threats and/or the levels of harmful effect to acceptable values.

The technical risks appear when the system cannot satisfy the system requirements any longer. The causes reside in the requirements and/or in the solution itself. They are expressed in the form of insufficient effectiveness and can have multiple causes: incorrect assessment of the technological capabilities; over-estimation of the technical maturity of a system element; failure of parts; breakdowns; breakage, obsolescence of equipment, parts, or software, weakness from the supplier (non-compliant parts, delay for supply, etc.), human factors (insufficient training, wrong tunings, error handling, unsuited procedures, malice), etc.

Technical risks are not to be confused with project risks, even if the method to manage them is the same. Although technical risks may lead to project risks, technical risks address the system itself, not the process for its development. (See Risk Management for more details.)

Process Approach

Purpose and Principles of the Approach

The system analysis process is used to: (1) provide a rigorous basis for technical decision making, resolution of requirement conflicts, and assessment of alternative physical solutions (system elements and physical architectures); (2) determine progress in satisfying system requirements and derived requirements; (3) support risk management; and (4) ensure that decisions are made only after evaluating the cost, schedule, performance, and risk effects on the engineering or re-engineering of a system (ANSI/EIA 1998). This process is also called the decision analysis process by NASA (2007, 1-360) and is used to help evaluate technical issues, alternatives, and their uncertainties to support decision-making. (See Decision Management for more details.)

System analysis supports other system definition processes:

- Stakeholder requirements definition and system requirements definition processes use system analysis to solve issues relating to conflicts among the set of requirements; in particular, those related to costs, technical risks, and effectiveness (performances, operational conditions, and constraints). System requirements subject to high risks, or those which would require different architectures, are discussed.
- The Logical Architecture Model Development and Physical Architecture Model Development processes use it to assess characteristics or design properties of candidate logical and physical architectures, providing arguments for selecting the most efficient one in terms of costs, technical risks, and effectiveness (e.g., performances, dependability, human factors, etc.).

Like any system definition process, the system analysis process is iterative. Each operation is carried out several times; each step improves the precision of analysis.

Activities of the Process

Major activities and tasks performed within this process include

- Planning the trade-off studies:
  - Determine the number of candidate solutions to analyze, the methods and procedures to be used, the expected results (examples of objects to be selected: behavioral architecture/scenario, physical architecture, system element, etc.), and the justification items.
  - Schedule the analyses according to the availability of models, engineering data (system requirements, design properties), skilled personnel, and procedures.
- Define the selection criteria model:
  - Select the assessment criteria from non-functional requirements (performances, operational conditions, constraints, etc.), and/or from design properties.
  - Sort and order the assessment criteria.
  - Establish a scale of comparison for each assessment criterion, and weigh every assessment criterion according to its level of relative importance with the others.
- Identify candidate solutions, related models, and data.
- Assess candidate solutions using previously defined methods or procedures:
  - Carry out costs analysis, technical risks analysis, and effectiveness analysis placing every candidate solution on every assessment criterion comparison scale.
  - Score every candidate solution as an assessment score.
- Provide results to the calling process: assessment criteria, comparison scales, solutions’ scores, assessment selection, and possibly recommendations and related arguments.

Artifacts and Ontology Elements

This process may create several artifacts, such as

- A selection criteria model (list, scales, weighing)
- Costs, risks, and effectiveness analysis reports
- Justification reports

This process handles the ontology elements of Table 2 within system analysis.

Table 2. Main Ontology Elements as Handled within System Analysis. (SEBoK Original)

Assessment Criterion	In the context of system analysis, an assessment criterion is a characteristic used to assess or compare system elements, physical interfaces, physical architectures, functional architectures/scenarios, or any engineering elements that can be compared.
	Identifier; name; description; relative weight; scalar weight
Assessment Selection	In the context of system analysis, an assessment selection is a technical management element based on an assessment score that justifies the selection of a system element, a physical interface, a physical architecture, or a functional architecture/scenario.
Assessment Score	In the context of system analysis, an assessment score is obtained assessing a system element, a physical interface, a physical architecture, a functional architecture/scenario using a set of assessment criteria.
	Identifier; name; description; value
Cost	In the context of systems engineering, a cost is an amount expressed in a given currency related to the value of a system element, a physical interface, and a physical architecture.
	Identifier; name; description; amount; type (development, production, utilization, maintenance, disposal); confidence interval; period of reference; estimation technique

Risk	An event having a probability of occurrence and consequences related to the system mission or on other characteristics. (Used for technical risk in engineering.). A risk is the combination of vulnerability a danger or threat.	
	Identifier; name description; status	

Checking Correctness of System Analysis

The main items to be checked within system analysis in order to get validated arguments are

- Relevance of the models and data in the context of use of the system,
- Relevance of assessment criteria related to the context of use of the system,
- Reproducibility of simulation results and of calculations,
- Precision level of comparisons' scales,
- Confidence of estimates, and
- Sensitivity of solutions' scores related to assessment criteria weights.

See Ring, Eisner, and Maier (2010) for additional perspective.

Methods and Modeling Techniques

- **General usage of models**: Various types of models can be used in the context of system analysis:
  - **Physical models** are scale models allowing simulation of physical phenomena. They are specific to each discipline; associated tools include mock-ups, vibration tables, test benches, prototypes, decompression chamber, wind tunnels, etc.
  - **Representation models** are mainly used to simulate the behavior of a system. For example, enhanced functional flow block diagrams (eFFBDs), statecharts, state machine diagrams (based in systems modeling language (SysML)), etc.
  - **Analytical models** are mainly used to establish values of estimates. We can consider the deterministic models and probabilistic models (also known as stochastic models) to be analytical in nature. Analytical models use equations or diagrams to approach the real operation of the system. They can be very simple (addition) to incredibly complicated (probabilistic distribution with several variables).
- **Use right models** depending on the project progress
  - At the beginning of the project, first studies use simple tools, allowing rough approximations which have the advantage of not requiring too much time and effort. These approximations are often sufficient to eliminate unrealistic or outgoing candidate solutions.
  - Progressively with the progress of the project it is necessary to improve precision of data to compare the candidate solutions still competing. The work is more complicated if the level of innovation is high.
  - A systems engineer alone cannot model a complex system; he has to be supported by skilled people from different disciplines involved.
- **Specialist expertise**: When the values of assessment criteria cannot be given in an objective or precise way, or because the subjective aspect is dominating, we can ask specialists for expertise. The estimates proceed in four steps:

1. Select interviewees to collect the opinion of qualified people for the considered field.
2. Draft a questionnaire; a precise questionnaire allows an easy analysis, but a questionnaire that is too closed risks the neglection of significant points.
3. Interview a limited number of specialists with the questionnaire, including an in-depth discussion to get precise opinions.
4. Analyze the data with several different people and compare their impressions until an agreement on a classification of assessment criteria and/or candidate solutions is reached.

Often used analytical models in the context of system analysis are summarized in Table 3.

Table 3. Often Used Analytical Models in the Context of System Analysis. (SEBoK Original)

Type of Model	Description
Deterministic models	• Models containing **statistics** are included in this category. The principle consists in establishing a model based on a significant amount of data and number of results from former projects; they can apply only to system elements/components whose technology already exists. • Models by **analogy** also use former projects. The system element being studied is compared to an already existing system element with known characteristics (cost, reliability, etc.). Then these characteristics are adjusted based on the specialists' expertise. • **Learning curves** allow foreseeing the evolution of a characteristic or a technology. One example of evolution: "Each time the number of produced units is multiplied by two, the cost of this unit is reduced with a certain percentage, generally constant."
Probabilistic models (also called stochastic models)	The theory of probability allows classifying the possible candidate solutions compared to consequences from a set of events as criteria. These models are applicable if the number of criteria is limited and the combination of the possible events is simple. Take care that the sum of probabilities of all events is equal to one for each node.
Multi-criteria decisions models	When the number of criteria is greater than ten, it is recommended that a multi-criteria decision model be established. This model is obtained through the following actions: • Organize the criteria as a hierarchy (or a decomposition tree). • Associate each criterion of each branch of the tree with a relative weight compare to each other of the same level. • Calculate a scalar weight for each leaf criterion of each branch multiplying all the weights of the branch. • Score every candidate solution on the leaf criteria; sum the scores to get a global score for each candidate solution; compare the scores. • Using a computerized tool allows to perform sensitivity analysis to get a robust choice.

Practical Considerations

Key pitfalls and good practices related to system analysis are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing system analysis are provided in Table 4.

Table 4. Pitfalls with System Analysis. (SEBoK Original)

Pitfall	Description
Analytical modeling is not a decision tool	Analytical modeling gives analytical results from analytical data. It has to be considered as a help and not as a decision tool.
Models and system levels of decomposition	A model can be well adapted to a level *n* of a system and to be incompatible with the model of the higher level which uses the data coming from the lower level. It is essential that the systems engineer ensures the coherence of the various models used.
Optimization is not a sum of optimized elements	The general optimization of the system-of-interest is not the sum of its optimized systems and/or system elements.

Proven Practices

Some proven practices gathered from the references are provided in Table 5.

Table 5. Proven Practices with System Analysis. (SEBoK Original)

Practice	Description
Stay in the operational field	Models can never simulate all the behavior/reactions of a system: they operate only in one limited field with a restricted number of variables. When a model is used, it is always necessary to make sure that the parameters and data inputs are part of the operation field. If not, there is a high risk of irregular outputs.
Evolve models	Models shall evolve during the project: by modification of parameter settings, by entering new data when modified (modification of assessment criteria, functions to perform, requirements, etc.), by the use of new tools when those used reach their limits.
Use several types of models	It is recommended to concurrently use several types of models in order to compare the results and/or to take into account another aspect of the system.
Keep context elements consistent	Results of a simulation shall always be given in their modeling context: tool used, selected assumptions, parameters and data introduced, and variance of the outputs.

References

Works Cited

ANSI/EIA. 1998. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA-632-1998.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Ring, J, H. Eisner, and M. Maier. 2010. "Key Issues of Systems Engineering, Part 3: Proving Your Design." INCOSE *Insight* 13(2).

Primary References

ANSI/EIA. 1998. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

Blanchard, B.S., and W.J. Fabrycky. 2010. *Systems Engineering and Analysis,* 5th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C., USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

Ring, J, H. Eisner, and M. Maier. 2010. "Key Issues of Systems Engineering, Part 3: Proving Your Design." INCOSE *Insight.* 13(2).

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: System Realization

System Realization

System realization activities are conducted to create and test versions of a system as specified by system definition. The activities are grouped and described as generic processes that are performed iteratively and/or concurrently depending on the selected Life Cycle Model (glossary). These activities include those required to build a system (system implementation), integrate disparate system elements (system integration), and ensure that the system meets both the needs of stakeholders (system validation) and aligns with the system requirements and architecture (system verification).

These activities are not sequential, but are performed concurrently, iteratively and recursively depending on the selected life cycle model. Figure 1 (see "Overview", below), also shows how these processes fit within the context of System Definition (glossary) and System Deployment and Use KAs. See also Applying Life Cycle Processes for further discussion of the relationships between process and life cycle model.

Topics

Each part of the SEBoK is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Implementation
- System Integration
- System Verification
- System Validation

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

Overview

Essentially, the outputs of system definition are used during system implementation to create system elements and during system integration to provide plans and criteria for combining these elements. The requirements are used to verify and validate system elements, systems, and the overall system-of-interest (SoI). These activities provide feedback into the system design, particularly when problems or challenges are identified.

Finally, when the system is considered, verified, and validated, it will then become an input to system deployment and use. It is important to understand that there is overlap in these activities; they do not have to occur in sequence as demonstrated in Figure 1. Every life cycle model includes realization activities, principally, verification and validation activities. The way these activities are performed is dependent upon the life cycle model in use. (For additional information on life cycles, see the Life Cycle Models KA.)

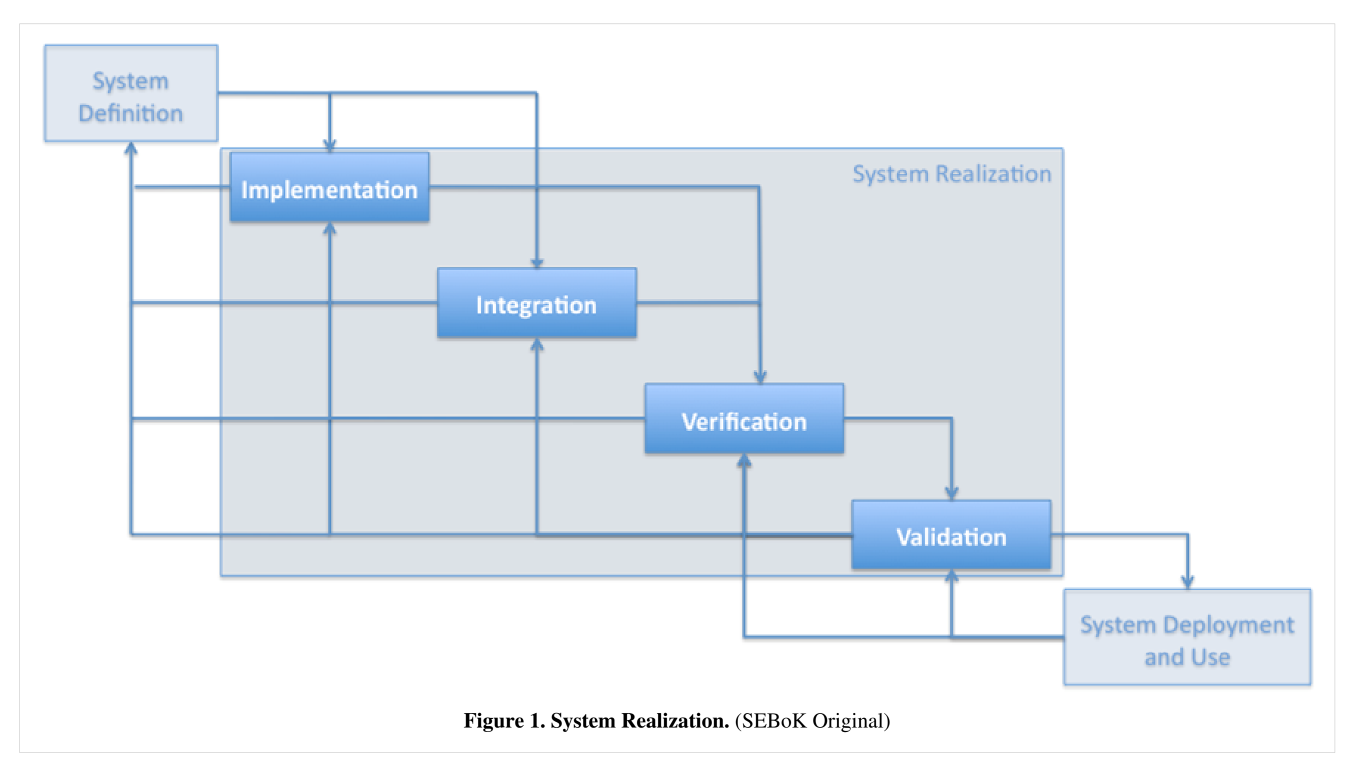


Figure 1. System Realization. (SEBoK Original)

The realization processes are performed to ensure that the system will be ready for transition and has the appropriate structure and behavior to enable the desired operation and functionality throughout the system's life span. Both DAU and NASA include transition in realization, in addition to implementation, integration, verification, and validation (Prosnik 2010; NASA December 2007, 1-360).

Fundamentals

Macro View of Realization Processes

Figure 2 illustrates a macro view of generic outputs from realization activities when using a Vee life cycle model. The left side of the Vee represents various design activities 'going down' the system.

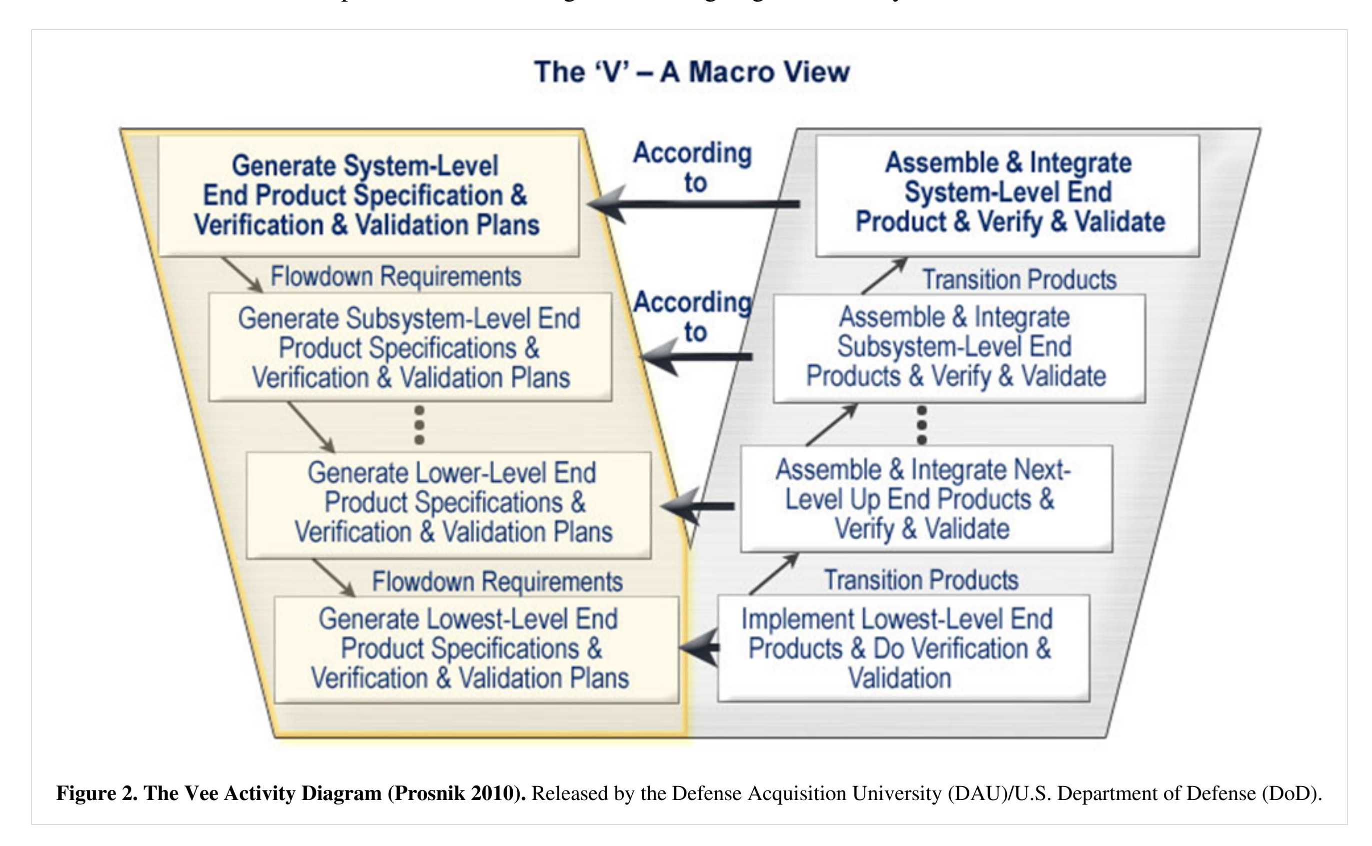


Figure 2. The Vee Activity Diagram (Prosnik 2010). Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

The left side of the Vee model demonstrates the development of system elements specifications and design descriptions. In this stage, verification and validation plans are developed, which are later used to determine whether realized system elements (products, services, or enterprises) are compliant with specifications and stakeholder requirements. Also, during this stage initial specifications become flow-down requirements for lower-level system models. In terms of time frame, these activities take place early in the system's life cycle. These activities are discussed further in the System Definition KA. However, it is important to understand that some of the system realization activities are initiated at the same time as system definition activities; this is the case with integration, verification and validation planning in particular.

The right side of the Vee model, as illustrated in Figure 2, shows the system elements (products, services, or enterprises) are assembled according to the system model described on the left side of the Vee (integration). Verification and validation activities determine how well the realized system fulfills the stakeholder requirements, the system requirements, and design properties. These activities should follow the plans developed on the left side of the Vee. Integration can be done continuously, incrementally and/or iteratively, supported by verification and validation (V&V) efforts. For example, integration typically starts at the bottom of the Vee and continues upwards to the top of the Vee.

The U.S. Defense Acquisition University (DAU) provides an overview of what occurs during system realization:

> *Once the products of all system models have been fully defined, Bottom-Up End Product Realization can be initiated. This begins by applying the Implementation Process to buy, build, code or reuse end products. These implemented end products are verified against their design descriptions and specifications, validated against Stakeholder Requirements and then transitioned to the next higher*

system model for integration. End products from the Integration Process are successively integrated upward, verified and validated, transitioned to the next acquisition phase or transitioned ultimately as the End Product to the user. (Prosnik 2010)

While the systems engineering (SE) technical processes are life cycle processes, the processes are concurrent, and the emphasis of the respective processes depends on the phase and maturity of the design. Figure 3 portrays (from left to right) a notional emphasis of the respective processes throughout the systems acquisition life cycle from the perspective of the U.S. Department of Defense (DoD). It is important to note that from this perspective, these processes do not follow a linear progression; instead, they are concurrent, with the amount of activity in a given area changing over the system's life cycle. The red boxes indicate the topics that will be discussed as part of realization.

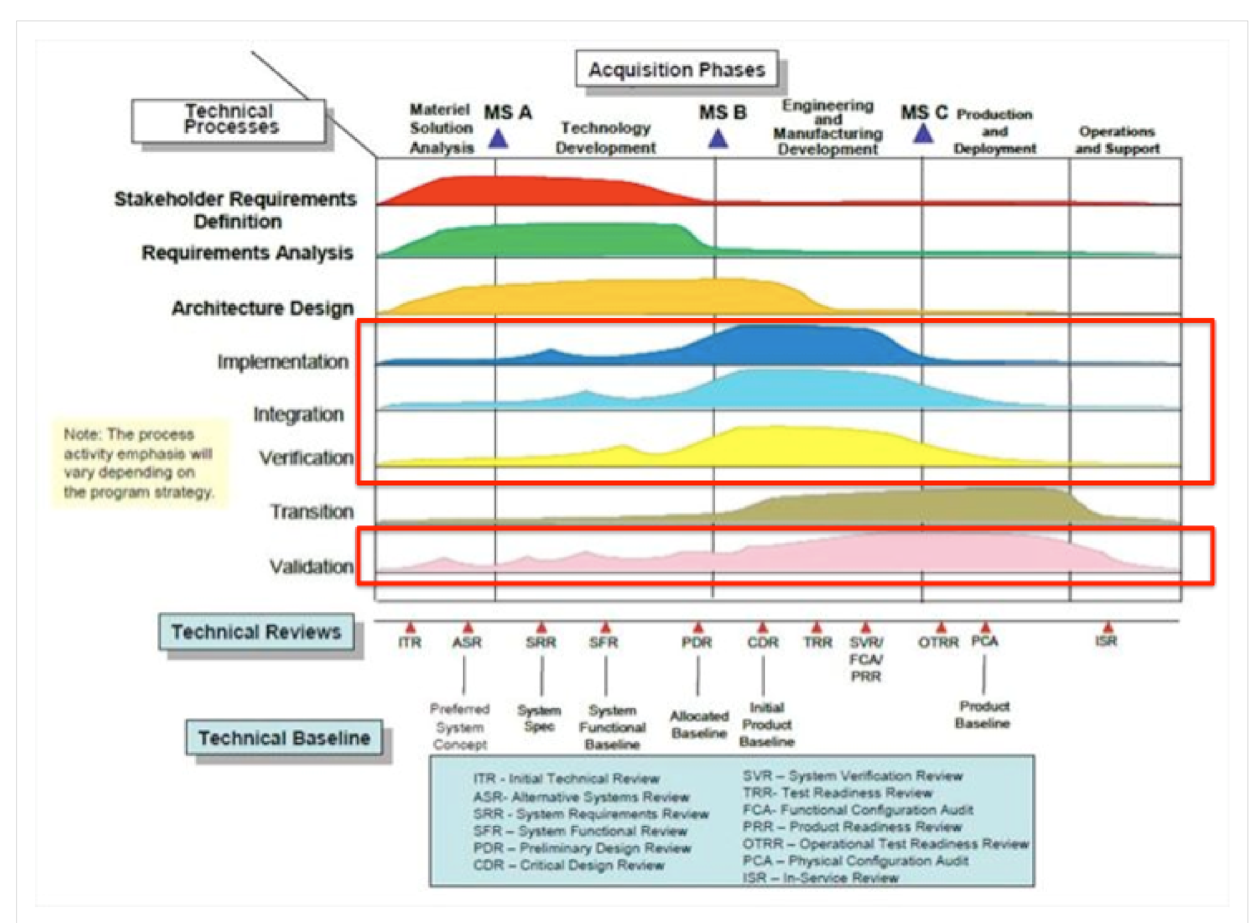


Figure 3. Notional Emphasis of Systems Engineering Technical Processes and Program Life-Cycle Phases (DAU 2010). Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

References

Works Cited

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Prosnik, G. 2010. Materials from "Systems 101: Fundamentals of Systems Engineering Planning, Research, Development, and Engineering". DAU distance learning program. eds. J. Snoderly, B. Zimmerman. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Primary References

INCOSE. 2011. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products,* 1st ed. Boca Raton, FL, USA: CRC Press.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

DAU. *Your Acquisition Policy and Discretionary Best Practices Guide.* In Defense Acquisition University (DAU)/U.S. Department of Defense (DoD) [database online]. Ft Belvoir, VA, USA. Available at: https://dag.dau.mil/Pages/Default.aspx (accessed 2010).

ECSS. 2009. *Systems Engineering General Requirements*. Noordwijk, Netherlands: Requirements and Standards Division, European Cooperation for Space Standardization (ECSS), 6 March 2009. ECSS-E-ST-10C.

IEEE. 2012. "Standard for System and Software Verification and Validation". Institute of Electrical and Electronics Engineers. IEEE-1012.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Implementation

System Implementation uses the structure created during architectural design and the results of system analysis to construct system elements that meet the stakeholder requirements and system requirements developed in the early life cycle phases. These system elements are then integrated to form intermediate aggregates and finally the complete system-of-interest (SoI). See System Integration.

Definition and Purpose

Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system breakdown structure). System elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing, the software realization processes of coding and testing, or the operational procedures development processes for operators' roles. If implementation involves a production process, a manufacturing system which uses the established technical and management processes may be required.

The purpose of the implementation process is to design and create (or fabricate) a system element conforming to that element's design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices. This process bridges the system definition processes and the integration process. Figure 1 portrays how the outputs of system definition relate to system implementation, which produces the implemented (system) elements required to produce aggregates and the SoI.

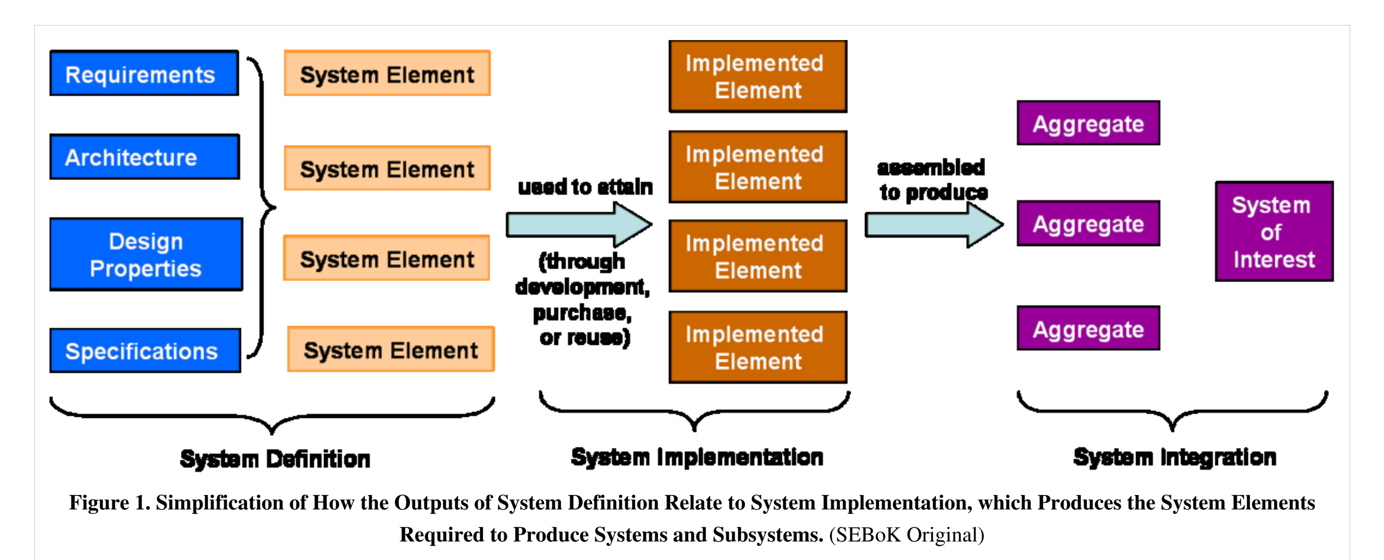


Figure 1. Simplification of How the Outputs of System Definition Relate to System Implementation, which Produces the System Elements Required to Produce Systems and Subsystems. (SEBoK Original)

Process Approach

Purpose and Principle of the Approach

During the implementation process, engineers apply the design properties and/or requirements allocated to a system element to design and produce a detailed description. They then fabricate, code, or build each individual element using specified materials, processes, physical or logical arrangements, standards, technologies, and/or information flows outlined in detailed descriptions (drawings or other design documentation). A system element will be verified against the detailed description of properties and validated against its requirements.

If subsequent verification and validation (V&V) actions or configuration audits reveal discrepancies, recursive interactions occur, which includes predecessor activities or processes, as required, to mitigate those discrepancies and to modify, repair, or correct the system element in question. Figure 2 provides the context for the implementation process from the perspective of the U.S. Defense Acquisition University (DAU).

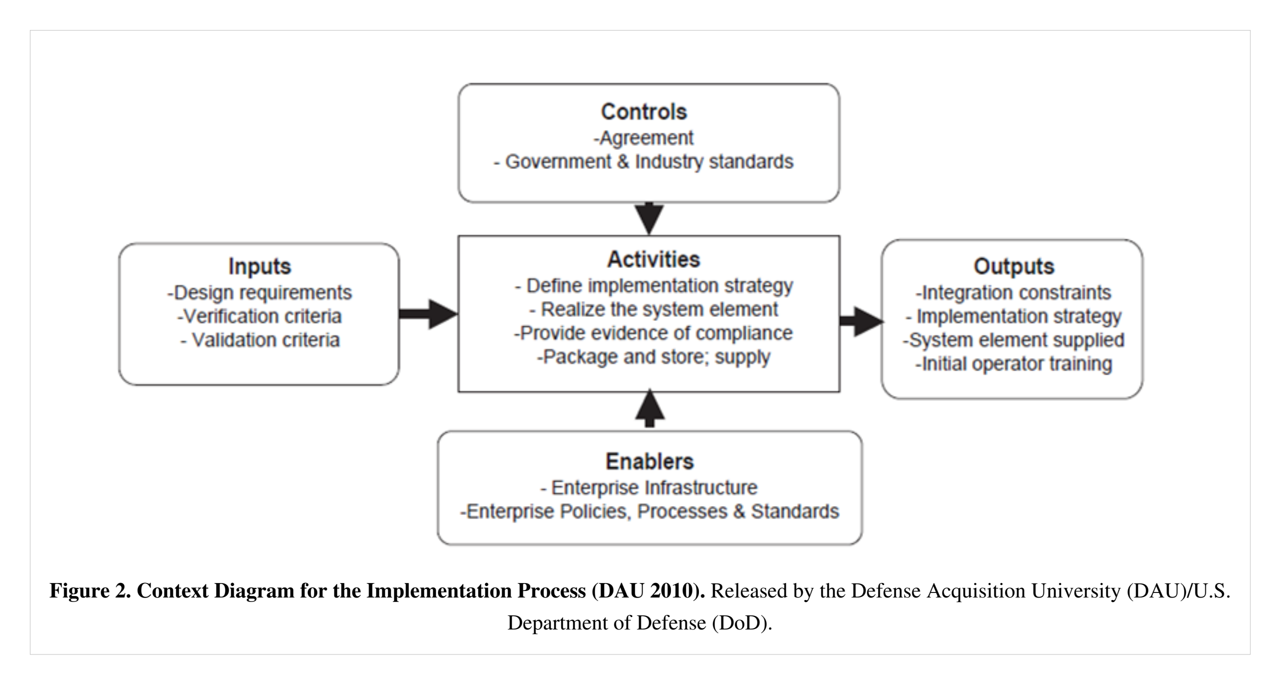


Figure 2. Context Diagram for the Implementation Process (DAU 2010). Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Such figures provide a useful overview of the systems engineering (SE) community's perspectives on what is required for implementation and what the general results of implementation may be. These are further supported by the discussion of implementation inputs, outputs, and activities found in the National Aeronautics and Space Association's (NASA's) *Systems Engineering Handbook* (NASA 2007). It is important to understand that these views are process-oriented. While this is a useful model, examining implementation only in terms of process can be limiting.

Depending on the technologies and systems chosen when a decision is made to produce a system element, the implementation process outcomes may generate constraints to be applied on the architecture of the higher-level system; those constraints are normally identified as derived system requirements and added to the set of system requirements applicable to this higher-level system. The architectural design has to take those constraints into account.

If the decision is made to purchase or reuse an existing system element, it has to be identified as a constraint or system requirement applicable to the architecture of the higher-level system. Conversely, the implementation process may involve some adaptation or adjustments to the system requirement in order to be integrated into a higher-level system or aggregate.

Implementation also involves packaging, handling, and storage, depending on the concerned technologies and where or when the system requirement needs to be integrated into a higher-level aggregate. Developing the supporting documentation for a system requirement, such as the manuals for operation, maintenance, and/or installation, is also a part of the implementation process; these artifacts are utilized in the system deployment and use phase. The system element requirements and the associated verification and validation criteria are inputs to this process; these inputs come from the architectural design process detailed outputs.

Execution of the implementation process is governed by both industrial and government standards and the terms of all applicable agreements. This may include conditions for packaging and storage, as well as preparation for use activities, such as operator training. In addition, packaging, handling, storage, and transportation (PHS&T) considerations will constrain the implementation activities. For more information, refer to the discussion of PHS&T in the System Deployment and Use article. The developing or integrating organization will likely have enterprise-level safety practices and guidelines that must also be considered.

Activities of the Process

The following major activities and tasks are performed during this process:

- **Define the implementation strategy** - Implementation process activities begin with detailed design and include developing an implementation strategy that defines fabrication and coding procedures, tools and equipment to be used, implementation tolerances, and the means and criteria for auditing configuration of resulting elements to the detailed design documentation. In the case of repeated system element implementations (such as for mass manufacturing or replacement elements), the implementation strategy is defined and refined to achieve consistent and repeatable element production; it is retained in the project decision database for future use. The implementation strategy contains the arrangements for packing, storing, and supplying the implemented element.
- **Realize the system element** - Realize or adapt and produce the concerned system element using the implementation strategy items as defined above. Realization or adaptation is conducted with regard to standards that govern applicable safety, security, privacy, and environmental guidelines or legislation and the practices of the relevant implementation technology. This requires the fabrication of hardware elements, development of software elements, definition of training capabilities, drafting of training documentation, and the training of initial operators and maintainers.
- **Provide evidence of compliance** - Record evidence that the system element meets its requirements and the associated verification and validation criteria as well as the legislation policy. This requires the conduction of peer reviews and unit testing, as well as inspection of operation and maintenance manuals. Acquire measured properties that characterize the implemented element (weight, capacities, effectiveness, level of performance, reliability, availability, etc.).
- **Package, store, and supply the implemented element** - This should be defined in the implementation strategy.

Artifacts and Ontology Elements

This process may create several artifacts such as

- an implemented system
- implementation tools
- implementation procedures
- an implementation plan or strategy
- verification reports
- issue, anomaly, or trouble reports
- change requests (about design)

This process handles the ontology elements shown in Table 1 below.

Table 1. Main Ontology Elements as Handled within System Element Implementation. (SEBoK Original)

Element	Definition Attributes (examples)
Implemented Element	An implemented element is a system element that has been implemented. In the case of hardware it is marked with a part/serial number. Identifier, name, description, type (hardware, software application, software piece, mechanical part, electric art, electronic component, operator role, procedure, protocol, manual, etc.)

Measured Property	A measured property is a characteristic of the implemented element established after its implementation. The measured properties characterize the implemented system element when it is completely realized, verified, and validated. If the implemented element complies with a design property, the measured property should equal the design property. Otherwise one has to identify the difference or non-conformance which treatment could conclude to modify the design property and possibly the related requirements, or to modify (correct, repair) the implemented element, or to identify a deviation.
	Identifier, name, description, type (effectiveness, availability, reliability, maintainability, weight, capacity, etc.), value, unit, etc.

The main relationships between ontology elements are presented in Figure 3.

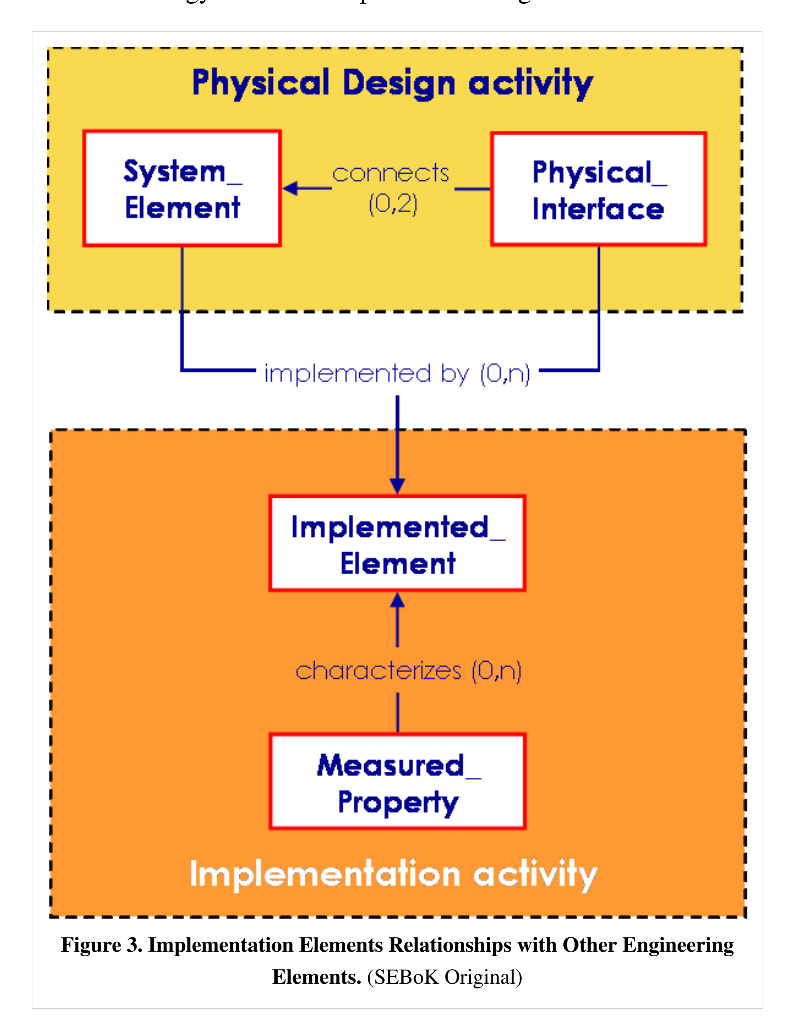


Figure 3. Implementation Elements Relationships with Other Engineering Elements. (SEBoK Original)

Methods, Techniques, and Tools

There are many software tools available in the implementation and integration phases. The most basic method would be the use of N-squared diagrams as discussed in Jeff Grady's book *System Integration* (Grady 1994).

Checking and Correctness of Implementation

Proper implementation checking and correctness should include testing to determine if the implemented element (i.e., piece of software, hardware, or other product) works in its intended use. Testing could include mockups and breadboards, as well as modeling and simulation of a prototype or completed pieces of a system. Once this is completed successfully, the next process would be system integration.

References

Works Cited

DAU. February 19, 2010. *Defense acquisition guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense.

Grady, J.O. 1994. *System integration.* Boca Raton, FL, USA: CRC Press, Inc.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Grady, J.O. 1994. *System Integration*. Boca Raton, FL, USA: CRC Press, Inc.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Integration

System integration consists of taking delivery of the implemented system elements which compose the system-of-interest (SoI), assembling these implemented elements together, and performing the verification and validation actions (V&V actions) in the course of the assembly. The ultimate goal of system integration is to ensure that the individual system elements function properly as a whole and satisfy the design properties or characteristics of the system. System integration is one part of the realization effort and relates only to developmental items. Integration should not to be confused with the assembly of end products on a production line. To perform the production, the assembly line uses a different order from that used by integration.

Definition and Purpose

System integration consists of a process that "*iteratively combines implemented system elements to form complete or partial system configurations in order to build a product or service. It is used recursively for successive levels of the system hierarchy*." (ISO/IEC 15288 2015, 68). The process is extended to any kind of product system, service system, and enterprise system. The purpose of system integration is to prepare the SoI for final validation and transition either for use or for production. Integration consists of progressively assembling aggregates of implemented elements that compose the SoI as architected during design, and to check correctness of static and dynamic aspects of interfaces between the implemented elements.

The U.S. Defense Acquisition University (DAU) provides the following context for integration: *The integration process will be used . . . for the incorporation of the final system into its operational environment to ensure that the system is integrated properly into all defined external interfaces. The interface management process is particularly important for the success of the integration process, and iteration between the two processes will occur* (DAU 2010).

The purpose of system integration can be summarized as below:

- Completely assemble the implemented elements to make sure that the they are compatible with each other.
- Demonstrate that the aggregates of implemented elements perform the expected functions and meet measures of performance/effectiveness.
- Detect defects/faults related to design and assembly activities by submitting the aggregates to focused V&V actions.

Note: In the systems engineering literature, sometimes the term *integration* is used in a larger context than in the present topic. In this larger sense, it concerns the technical effort to simultaneously design and develop the system and the processes for developing the system through concurrent consideration of all life cycle stages, needs, and competences. This approach requires the "integration" of numerous skills, activities, or processes.

Principles

Boundary of Integration Activity

Integration can be understood as the whole bottom-up branch of the Vee Model, including the tasks of assembly and the appropriate verification tasks. See Figure 1 below:

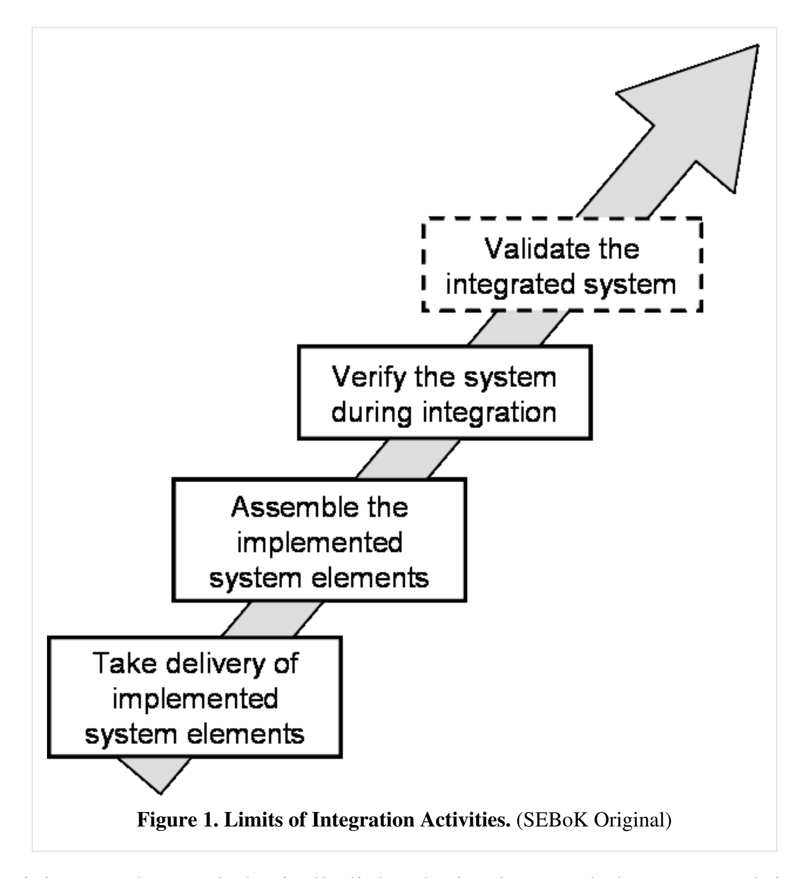


Figure 1. Limits of Integration Activities. (SEBoK Original)

The assembly activity joins together, and physically links, the implemented elements. Each implemented element is individually verified and validated prior to entering integration. Integration then adds the verification activity to the assembly activity, excluding the final validation.

The final validation performs operational tests that authorize the transition for use or the transition for production. Remember that system integration only endeavors to obtain pre-production prototypes of the concerned product, service, or enterprise. If the product, service, or enterprise is delivered as a unique exemplar, the final validation activity serves as acceptance for delivery and transfer for use. If the prototype has to be produced in several exemplars, the final validation serves as acceptance to launch their production. The definition of the optimized operations of assembly which will be carried out on a production line relates to the manufacturing process and not to the integration process.

Integration activity can sometimes reveal issues or anomalies that require modifications of the design of the system. Modifying the design is not part of the integration process but concerns only the design process. Integration only deals with the assembly of the implemented elements and verification of the system against its properties as designed. During assembly, it is possible to carry out tasks of finishing touches which require simultaneous use of several implemented elements (e.g., paint the whole after assembly, calibrate a biochemical component, etc.). These tasks must be planned in the context of integration and are not carried out on separate implemented elements and do not include modifications related to design.

Aggregation of Implemented Elements

The integration is used to systematically assemble a higher-level system from lower-level ones (implemented system elements) that have been implemented. Integration often begins with analysis and simulations (e.g., various types of prototypes) and progresses through increasingly more realistic systems and system elements until the final product, service, or enterprise is achieved.

System integration is based on the notion of an aggregate - a subset of the system made up of several implemented elements (implemented system elements and physical interfaces) on which a set of V&V actions is applied. Each aggregate is characterized by a configuration which specifies the implemented elements to be physically assembled and their configuration status.

To perform V&V actions, a V&V configuration that includes the aggregate plus V&V tools is constituted. The V&V tools are enabling products and can be simulators (simulated implemented elements), stubs or caps, activators (launchers, drivers), harness, measuring devices, etc.

Integration by Level of System

According to the Vee Model, system definition (top-down branch) is done by successive levels of decomposition; each level corresponds to the physical architecture of systems and system elements. The integration (bottom-up branch) takes the opposite approach of composition (i.e., a level by level approach). On a given level, integration is done on the basis of the physical architecture defined during system definition.

Integration Strategy

The integration of implemented elements is generally performed according to a predefined strategy. The definition of the integration strategy is based on the architecture of the system and relies on the way the architecture of the system has been designed. The strategy is described in an integration plan that defines the minimum configuration of expected aggregates, the order of assembly of these aggregates in order to support efficient subsequent verification and validation actions (e.g., inspections and/or testing), techniques to check or evaluate interfaces, and necessary capabilities in the integration environment to support combinations of aggregates. The integration strategy is thus elaborated starting from the selected verification and validation strategy. See the System Verification and System Validation topics.

To define an integration strategy, there are several possible integration approaches/techniques that may be used individually or in combination. The selection of integration techniques depends on several factors; in particular, the type of system element, delivery time, order of delivery, risks, constraints, etc. Each integration technique has strengths and weaknesses which should be considered in the context of the SoI. Some integration techniques are summarized in Table 1 below.

Table 1. Integration Techniques. (SEBoK Original)

Integration Technique	Description
Global Integration	Also known as *big-bang integration*; all the delivered implemented elements are assembled in only one step. • This technique is simple and does not require simulating the implemented elements not being available at that time. • Difficult to detect and localize faults; interface faults are detected late. • Should be reserved for simple systems, with few interactions and few implemented elements without technological risks.

Integration "with the Stream"	The delivered implemented elements are assembled as they become available. • Allows starting the integration quickly. • Complex to implement because of the necessity to simulate the implemented elements not yet available. Impossible to control the end-to-end "functional chains"; consequently, global tests are postponed very late in the schedule. • Should be reserved for well known and controlled systems without technological risks.
Incremental Integration	In a predefined order, one or a very few implemented elements are added to an already integrated increment of implemented elements. • Fast localization of faults: a new fault is usually localized in lately integrated implemented elements or dependent of a faulty interface. • Require simulators for absent implemented elements. Require many test cases, as each implemented element addition requires the verification of the new configuration and regression testing. • Applicable to any type of architecture.
Subsets Integration	Implemented elements are assembled by subsets, and then subsets are assembled together (a subset is an aggregate); could also be called "functional chains integration". • Time saving due to parallel integration of subsets; delivery of partial products is possible. Requires less means and fewer test cases than integration by increments. • Subsets shall be defined during the design. • Applicable to architectures composed of sub-systems.
Top-Down Integration	Implemented elements or aggregates are integrated in their activation or utilization order. • Availability of a skeleton and early detection of architectural faults, definition of test cases close to reality, and the re-use of test data sets possible. • Many stubs/caps need to be created; difficult to define test cases of the leaf-implemented elements (lowest level). • Mainly used in software domain. Start from the implemented element of higher level; implemented elements of lower level are added until leaf-implemented elements.
Bottom-Up Integration	Implemented elements or aggregates are integrated in the opposite order of their activation or utilization. • Easy definition of test cases - early detection of faults (usually localized in the leaf-implemented elements); reduce the number of simulators to be used. An aggregate can be a sub-system. • Test cases shall be redefined for each step, drivers are difficult to define and realize, implemented elements of lower levels are "over-tested", and does not allow to quickly detecting the architectural faults. • Mainly used in software domain and in any kind of system.
Criterion Driven Integration	The most critical implemented elements compared to the selected criterion are first integrated (dependability, complexity, technological innovation, etc.). Criteria are generally related to risks. • Allow testing early and intensively critical implemented elements; early verification of design choices. • Test cases and test data sets are difficult to define.

Usually, a mixed integration technique is selected as a trade-off between the different techniques listed above, allowing optimization of work and adaptation of the process to the system under development. The optimization takes into account the realization time of the implemented elements, their delivery scheduled order, their level of complexity, the technical risks, the availability of assembly tools, cost, deadlines, specific personnel capability, etc.

Process Approach

Activities of the Process

Major activities and tasks performed during this process include

- **Establishing the integration plan** (this activity is carried out concurrently to the design activity of the system) that defines:
  - The optimized integration strategy: order of aggregates assembly using appropriate integration techniques.
  - The V&V actions to be processed for the purpose of integration.
  - The configurations of the aggregates to be assembled and verified.
  - The integration means and verification means (dedicated enabling products) that may include assembly procedures, assembly tools (harness, specific tools), V&V tools (simulators, stubs/caps, launchers, test benches, devices for measuring, etc.), and V&V procedures.
- **Obtain the integration means** and verification means as defined in the integration plan; the acquisition of the means can be done through various ways such as procurement, development, reuse, and sub-contracting; usually the acquisition of the complete set of means is a mix of these methods.
- **Take delivery** of each implemented element:
  - Unpack and reassemble the implemented element with its accessories.
  - Check the delivered configuration, conformance of implemented elements, compatibility of interfaces, and ensure the presence of mandatory documentation.
- **Assemble the implemented elements** into aggregates:
  - Gather the implemented elements to be assembled, the integration means (assembly tools, assembly procedures), and the verification means (V&V tools and procedures).
  - Connect the implemented elements on each other to constitute aggregates in the order prescribed by the integration plan and in assembly procedures using assembly tools.
  - Add or connect the V&V tools to the aggregates as predefined.
  - Carry out eventual operations of welding, gluing, drilling, tapping, adjusting, tuning, painting, parametering, etc.
- **Verify each aggregate**:
  - Check the aggregate is correctly assembled according to established procedures.
  - Perform the verification process that uses verification and validation procedures and check that the aggregate shows the right design properties/specified requirements.
  - Record integration results/reports and potential issue reports, change requests, etc.

Artifacts and Ontology Elements

This process may create several artifacts such as

- an integrated system
- assembly tools
- assembly procedures
- integration plans
- integration reports
- issue/anomaly/trouble reports
- change requests (about design)

This process utilizes the ontology elements discussed in Table 2.

Table 2. Main Ontology Elements as Handled within System Integration. (SEBoK Original)

Element	Definition	Attributes
Aggregate	An aggregate is a subset of the system made up of several system elements or systems on which a set of verification actions is applied.	Identifier, name, description
Assembly Procedure	An assembly procedure groups a set of elementary assembly actions to build an aggregate of implemented system elements.	Identifier, name, description, duration, unit of time
Assembly Tool	An assembly tool is a physical tool used to connect, assemble, or link several implemented system elements to build aggregates (specific tool, harness, etc.).	Identifier, name, description
Risk	An event having a probability of occurrence and a gravity degree on its consequence onto the system mission or on other characteristics (used for technical risk in engineering). A risk is the combination of vulnerability and of a danger or a threat.	Identifier, name, description, status
Rationale	An argument that provides the justification for the selection of an engineering element.	Identifier, name, description (rational, reasons for defining an aggregate, assembly procedure, assembly tool)

Note: verification and validation ontology elements are described in the System Verification and System Validation topics.

The main relationships between ontology elements are presented in Figure 2.

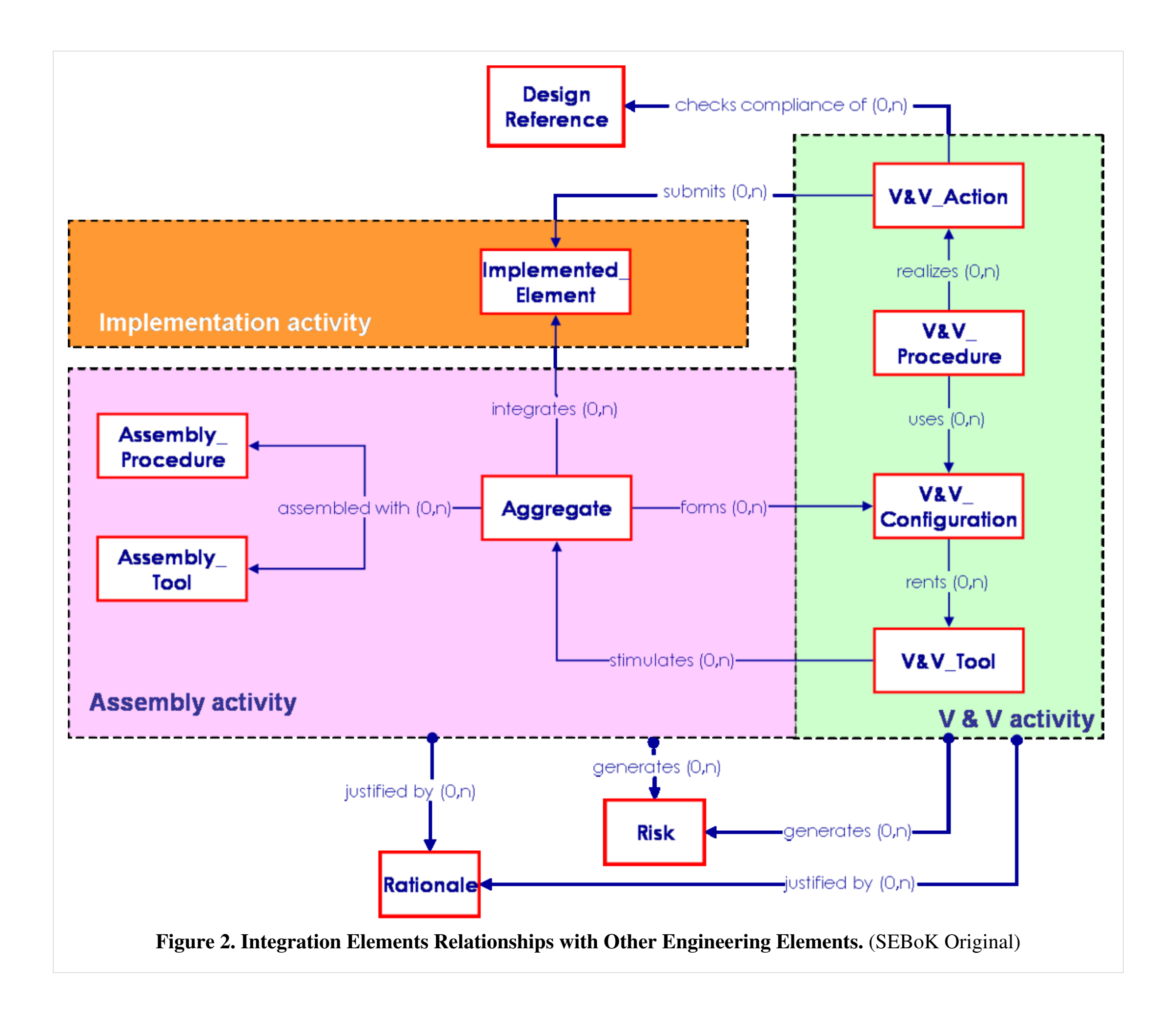


Figure 2. Integration Elements Relationships with Other Engineering Elements. (SEBoK Original)

Checking and Correctness of Integration

The main items to be checked during the integration process include the following:

- The integration plan respects its template.
- The expected assembly order (integration strategy) is realistic.
- No system element and physical interface set out in the system design document is forgotten.
- Every interface and interaction between implemented elements is verified.
- Assembly procedures and assembly tools are available and validated prior to beginning the assembly.
- V&V procedures and tools are available and validated prior to beginning the verification.
- Integration reports are recorded.

Methods and Techniques

Several different approaches are summarized above in the section Integration Strategy [1] (above) that may be used for integration, yet other approaches exist. In particular, important integration strategies for intensive software systems include: vertical integration, horizontal integration, and star integration.

Coupling Matrix and N-squared Diagram

One of the most basic methods to define the aggregates and the order of integration would be the use of N-Squared diagrams (Grady 1994, 190).

In the integration context, the coupling matrices are useful for optimizing the aggregate definition and verification of interfaces:

- The integration strategy is defined and optimized by reorganizing the coupling matrix in order to group the implemented elements in aggregates, thus minimizing the number of interfaces to be verified between aggregates (see Figure 3).

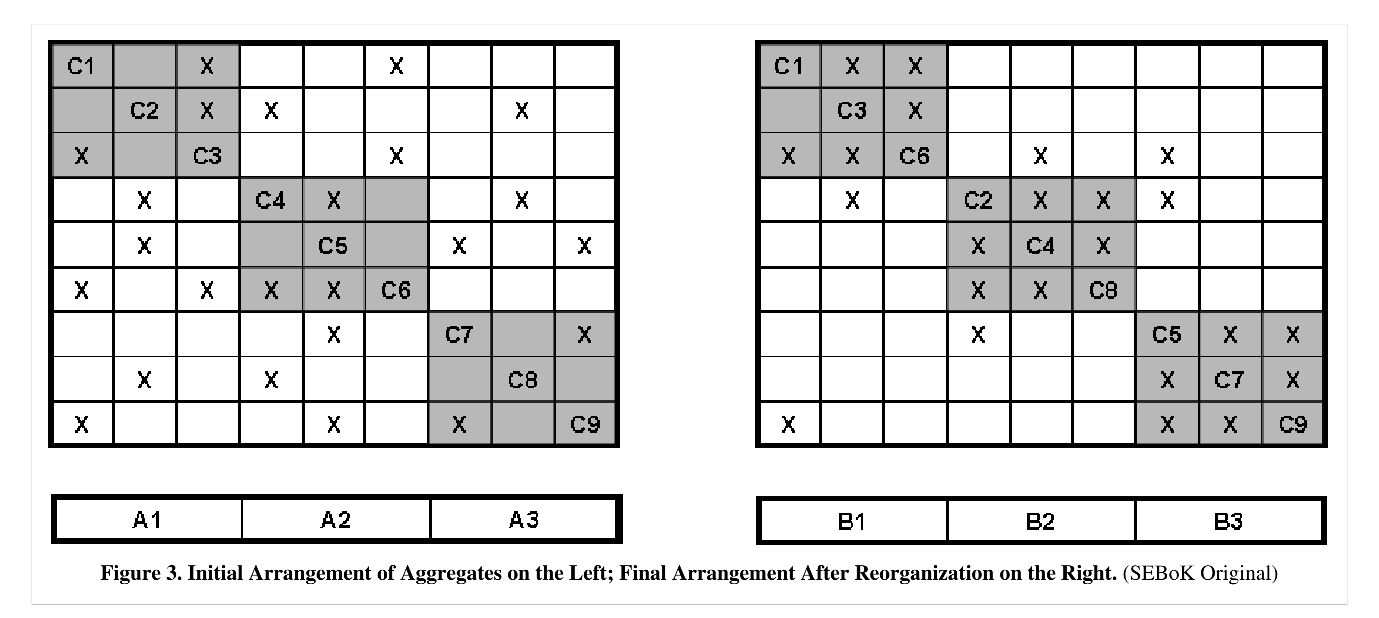


Figure 3. Initial Arrangement of Aggregates on the Left; Final Arrangement After Reorganization on the Right. (SEBoK Original)

- When verifying the interactions between aggregates, the matrix is an aid tool for fault detection. If by adding an implemented element to an aggregate an error is detected, the fault can be either related to the implemented element, to the aggregate, or to the interfaces. If the fault is related to the aggregate, it can relate to any implemented element or any interface between the implemented elements internal to the aggregate.

Application to Product Systems, Service Systems, and Enterprise Systems

As the nature of implemented system elements and physical interfaces is different for these types of systems, the aggregates, the assembly tools, and the V&V tools are different. Some integration techniques are more appropriate to specific types of systems. Table 3 below provides some examples.

Table 3. Different Integration Elements for Product, Service, and Enterprise Systems. (SEBoK Original)

Element	Product System	Service System	Enterprise System
System Element	Hardware Parts (mechanics, electronics, electrical, plastic, chemical, etc.) Operator Roles Software Pieces	Processes, data bases, procedures, etc. Operator Roles Software Applications	Corporate, direction, division, department, project, technical team, leader, etc. IT components
Physical Interface	Hardware parts, protocols, procedures, etc.	Protocols, documents, etc.	Protocols, procedures, documents, etc.
Assembly Tools	Harness, mechanical tools, specific tools Software Linker	Documentation, learning course, etc.	Documentation, learning, moving of office
Verification Tools	Test bench, simulator, launchers, stub/cap	Activity/scenario models, simulator, human roles rehearsal, computer, etc. Skilled Experts	Activity/scenario models, simulator, human roles rehearsal
Validation Tools	Operational environment	Operational environment	Operational environment

Recommended Integration Techniques	Top down integration technique Bottom Up Integration technique	Subsets integration technique (functional chains)	Global integration technique Incremental integration

Practical Considerations

Key pitfalls and good practices related to system integration are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing SE Measurement are provided in Table 4.

Table 4. Major Pitfalls with System Integration. (SEBoK Original)

Pitfall	Description
What is expected has delay	The experience shows that the implemented elements always do not arrive in the expected order and the tests never proceed or result as foreseen; therefore, the integration strategy should allow a great flexibility.
Big-bang not appropriate	The "big-bang" integration technique is not appropriate for a fast detection of faults. It is thus preferable to verify the interfaces progressively all along the integration.
Integration plan too late	The preparation of the integration activities is planned too late in the project schedule, typically when first implemented elements are delivered.

Good Practices

Some good practices, gathered from the references are provided in Table 5.

Table 5. Proven Practices with System Integration. (SEBoK Original)

Practice	Description
Start earlier development of means	The development of assembly tools and verification and validation tools can be as long as the system itself. It should be started as early as possible as soon as the preliminary design is nearly frozen.
Integration means seen as enabling systems	The development of integration means (assembly tools, verification, and validation tools) can be seen as enabling systems, using system definition and system realization processes as described in this SEBoK, and managed as projects. These projects can be led by the project of the corresponding system-of-interest, but assigned to specific system blocks, or can be subcontracted as separate projects.
Use coupling matrix	A good practice consists in gradually integrating aggregates in order to detect faults more easily. The use of the coupling matrix applies for all strategies and especially for the bottom up integration strategy.
Flexible integration plan and schedule	The integration process of complex systems cannot be easily foreseeable and its progress control difficult to observe. This is why it is recommended to plan integration with specific margins, using flexible techniques, and integrating sets by similar technologies.
Integration and design teams	The integration responsible should be part of the design team.

References

Works Cited

DAU. February 19, 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

Primary References

INCOSE. 2010. *Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities*. Version 3.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods.* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense. February 19, 2010.

Gold-Bernstein, B. and W.A. Ruh. 2004. *Enterprise integration: The essential guide to integration solutions*. Boston, MA, USA: Addison Wesley Professional.

Grady, J.O. 1994. *System integration*. Boca Raton, FL, USA: CRC Press, Inc.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight* 12(4):59-63.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions*. Hoboken, NJ, USA: John Wiley & Sons.

Reason, J. 1997. *Managing the Risks of Organizational Accidents.* Aldershot, UK: Ashgate Publishing Limited.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://sebokwiki.org/1.0.1/index.php?title=System_Integration#Integration_Strategy

System Verification

System Verification is a set of actions used to check the *correctness* of any element, such as a system element, a system, a document, a service, a task, a requirement, etc. These types of actions are planned and carried out throughout the life cycle of the system. Verification is a generic term that needs to be instantiated within the context it occurs. As a process, verification is a transverse activity to every life cycle stage of the system. In particular, during the development cycle of the system, the verification process is performed in parallel with the system definition and system realization processes and applies to any activity and any product resulting from the activity. The activities of every life cycle process and those of the verification process can work together. For example, the integration process frequently uses the verification process. It is important to remember that verification, while separate from validation, is intended to be performed in conjunction with validation.

Definition and Purpose

Verification is the confirmation, through the provision of objective evidence, that specified requirements have been fulfilled. With a note added in ISO/IEC/IEEE 15288, the scope of verification includes a set of activities that compares a system or system element against the requirements, architecture and design characteristics, and other properties to be verified (ISO/IEC/IEEE 2015). This may include, but is not limited to, specified requirements, design description, and the system itself.

The purpose of verification, as a generic action, is to identify the faults/defects introduced at the time of any transformation of inputs into outputs. Verification is used to provide information and evidence that the transformation was made according to the selected and appropriate methods, techniques, standards, or rules.

Verification is based on tangible evidence; i.e., it is based on information whose veracity can be demonstrated by factual results obtained from techniques such as inspection, measurement, testing, analysis, calculation, etc. Thus, the process of verifying a system (product, service, enterprise, or system of systems (SoS)) consists of comparing the realized characteristics or properties of the product, service, or enterprise against its expected design properties.

Principles and Concepts

Concept of Verification Action

Why Verify?

In the context of human realization, any human thought is susceptible to error. This is also the case with any engineering activity. Studies in human reliability have shown that people trained to perform a specific operation make around 1-3 errors per hour in best case scenarios. In any activity, or resulting outcome of an activity, the search for potential errors should not be neglected, regardless of whether or not one thinks they will happen or that they should not happen; the consequences of errors can cause extremely significant failures or threats.

A **verification action** is defined, and then performed, as shown in Figure 1.

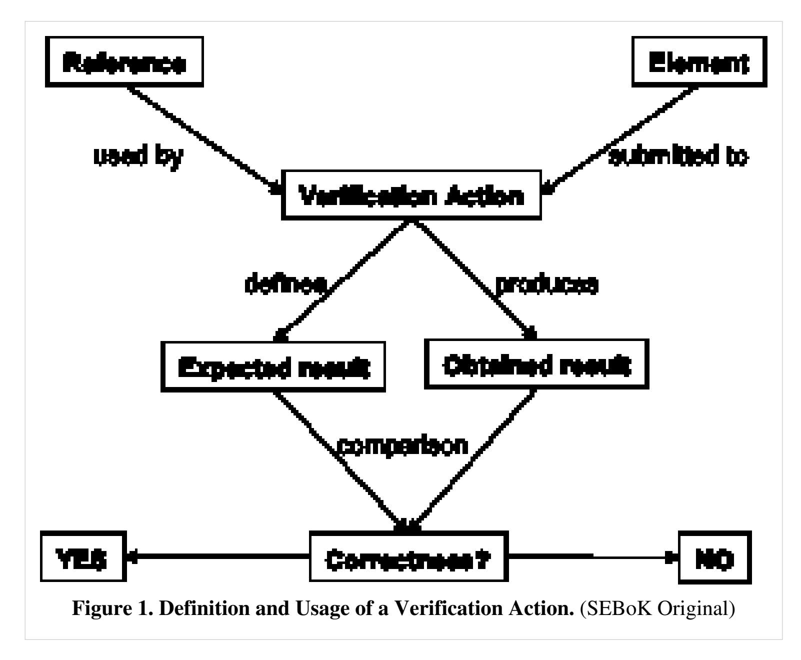


Figure 1. Definition and Usage of a Verification Action. (SEBoK Original)

The definition of a verification action applied to an engineering element includes the following:

- Identification of the element on which the verification action will be performed
- Identification of the reference to define the expected result of the verification action (see examples of reference in Table 1)

The performance of a verification action includes the following:

- Obtaining a result by performing the verification action onto the submitted element
- Comparing the obtained result with the expected result
- Deducing the degree of correctness of the element

What to Verify?

Any engineering element can be verified using a specific reference for comparison: stakeholder requirement, system requirement, function, system element, document, etc. Examples are provided in Table 1.

Table 1. Examples of Verified Items. (SEBoK Original)

Items	Explanation for Verification
Document	To verify a document is to check the application of drafting rules.
Stakeholder Requirement and System Requirement	To verify a stakeholder requirement or a system requirement is to check the application of syntactic and grammatical rules, and characteristics defined in the stakeholder requirements definition process, and the system requirements definition process such as; necessity, implementation free, unambiguous, consistent, complete, singular, feasible, traceable, and verifiable.
Design	To verify the design of a system is to check its logical and physical architecture elements against the characteristics of the outcomes of the design processes.
System	To verify a system (product, service, or enterprise) is to check its realized characteristics or properties against its expected design characteristics.
Aggregate	To verify an aggregate for integration is to check in particular every interface and interaction between implemented elements.
Verification Procedure	To verify a verification procedure is to check the application of a predefined template and drafting rules.

Verification versus Validation

The term *verification* is often associated with the term *validation* and understood as a single concept of *V&V*. Validation is used to ensure that *one is working the right problem*, whereas verification is used to ensure that *one has solved the problem right* (Martin 1997). From an actual and etymological meaning, the term verification comes from the Latin *verus*, which means truth, and *facere*, which means to make/perform. Thus, verification means to prove that something is *true* or correct (a property, a characteristic, etc.). The term validation comes from the Latin *valere*, which means to become strong, and has the same etymological root as the word *value*. Thus, validation means to prove that something has the right features to produce the expected effects. (Adapted from "Verification and Validation in plain English" (Lake INCOSE 1999).)

The main differences between the verification process and the validation process concern the references used to check the correctness of an element, and the acceptability of the effective correctness.

- Within verification, comparison between the expected result and the obtained result is generally binary, whereas within validation, the result of the comparison may require a judgment of value regarding whether or not to accept the obtained result compared to a threshold or limit.
- Verification relates more to one element, whereas validation relates more to a set of elements and considers this set as a whole.
- Validation presupposes that verification actions have already been performed.
- The techniques used to define and perform the verification actions and those for validation actions are very similar.

Integration, Verification, and Validation of the System

There is sometimes a misconception that verification occurs after integration and before validation. In most cases, it is more appropriate to begin verification activities during development or implementation and to continue them into deployment and use.

Once the system elements have been realized, they're integrated to form the complete system. Integration consists of assembling and performing verification actions as stated in the integration process. A final validation activity generally occurs when the system is integrated, but a certain number of validation actions are also performed parallel to the system integration in order to reduce the number of verification actions and validation actions while controlling the risks that could be generated if some checks are excluded. Integration, verification, and validation are intimately processed together due to the necessity of optimizing the strategy of verification and validation, as well as the strategy of integration.

Process Approach

Purpose and Principle of the Approach

The purpose of the verification process is to confirm that the system fulfills the specified design requirements. This process provides the information required to effect the remedial actions that correct non-conformances in the realized system or the processes that act on it - see ISO/IEC/IEEE 15288 (ISO/IEC/IEEE 2015).

Each system element and the complete system itself should be compared against its own design references (specified requirements). As stated by Dennis Buede, *verification is the matching of [configuration items], components, sub-systems, and the system to corresponding requirements to ensure that each has been built right* (Buede 2009). This means that the verification process is instantiated as many times as necessary during the global development of the system. Because of the generic nature of a process, the verification process can be applied to any engineering element that has conducted to the definition and realization of the system elements and the system itself.

Facing the huge number of potential verification actions that may be generated by the normal approach, it is necessary to optimize the verification strategy. This strategy is based on the balance between what must be verified and constraints, such as time, cost, and feasibility of testing, which naturally limit the number of verification actions and the risks one accepts when excluding some verification actions.

Several approaches exist that may be used for defining the verification process. The International Council on Systems Engineering (INCOSE) dictates that two main steps are necessary for verification: planning and performing verification actions (INCOSE 2012). NASA has a slightly more detailed approach that includes five main steps: prepare verification, perform verification, analyze outcomes, produce a report, and capture work products (NASA December 2007, 1-360, p. 102). Any approach may be used, provided that it is appropriate to the scope of the system, the constraints of the project, includes the activities of the process listed below in some way, and is appropriately coordinated with other activities.

Generic inputs are baseline references of the submitted element. If the element is a system, inputs are the logical and physical architecture elements as described in a system design document, the design description of internal interfaces to the system and interfaces requirements external to the system, and by extension, the system requirements.

Generic outputs define the verification plan that includes verification strategy, selected verification actions, verification procedures, verification tools, the verified element or system, verification reports, issue/trouble reports, and change requests on design.

Activities of the Process

To establish the verification strategy drafted in a verification plan (this activity is carried out concurrently to system definition activities), the following steps are necessary:

- Identify verification scope by listing as many characteristics or properties as possible that should be checked. The number of verification actions can be extremely high.
- Identify constraints according to their origin (technical feasibility, management constraints as cost, time, availability of verification means or qualified personnel, and contractual constraints that are critical to the mission) that limit potential verification actions.
- Define appropriate verification techniques to be applied, such as inspection, analysis, simulation, peer-review, testing, etc., based on the best step of the project to perform every verification action according to the given constraints.
- Consider a tradeoff of what should be verified (scope) taking into account all constraints or limits and deduce what can be verified; the selection of verification actions would be made according to the type of system, objectives of the project, acceptable risks, and constraints.
- Optimize the verification strategy by defining the most appropriate verification technique for every verification action while defining necessary verification means (tools, test-benches, personnel, location, and facilities) according to the selected verification technique.
- Schedule the execution of verification actions in the project steps or milestones and define the configuration of elements submitted to verification actions (this mainly involves testing on physical elements).

Performing verification actions includes the following tasks:

- Detail each verification action; in particular, note the expected results, the verification techniques to be applied, and the corresponding means required (equipment, resources, and qualified personnel).
- Acquire verification means used during system definition steps (qualified personnel, modeling tools, mocks-up, simulators, and facilities), and then those used during the integration step (qualified personnel, verification tools, measuring equipment, facilities, verification procedures, etc.).
- Carry out verification procedures at the right time, in the expected environment, with the expected means, tools, and techniques.

- Capture and record the results obtained when performing verification actions using verification procedures and means.

The obtained results must be analyzed and compared to the expected results so that the status may be recorded as either *compliant* or *non-compliant*. Systems engineering (SE) practitioners will likely need to generate verification reports, as well as potential issue/trouble reports, and change requests on design as necessary.

Controlling the process includes the following tasks:

- Update the verification plan according to the progress of the project; in particular, planned verification actions can be redefined because of unexpected events.
- Coordinate verification activities with the project manager: review the schedule and the acquisition of means, personnel, and resources. Coordinate with designers for issues/trouble/non-conformance reports and with the configuration manager for versions of the physical elements, design baselines, etc.

Artifacts and Ontology Elements

This process may create several artifacts such as:

- verification plans (contain the verification strategy)
- verification matrices (contain the verification action, submitted element, applied technique, step of execution, system block concerned, expected result, obtained result, etc.)
- verification procedures (describe verification actions to be performed, verification tools needed, the verification configuration, resources and personnel needed, the schedule, etc.)
- verification reports
- verification tools
- verified elements
- issue / non-conformance / trouble reports
- change requests to the design

This process utilizes the ontology elements displayed in Table 2 below.

Table 2. Main Ontology Elements as Handled within Verification. (SEBoK Original)

Element	Definition Attributes (examples)
Verification Action	A verification action describes what must be verified (the element as reference), on which element, the expected result, the verification technique to apply, on which level of decomposition. Identifier, name, description
Verification Procedure	A verification procedure groups a set of verification actions performed together (as a scenario of tests) in a gin verification configuration. Identifier, name, description, duration, unit of time
Verification Tool	A verification tool is a device or physical tool used to perform verification procedures (test bench, simulator, cap/stub, launcher, etc.). Identifier, name, description
Verification Configuration	A verification configuration groups all physical elements (aggregates and verification tools) necessary to perform a verification procedure. Identifier, name, description

Risk	An event having a probability of occurrence and a gravity degree on its consequence onto the system mission or on other characteristics (used for technical risk in engineering). A risk is the combination of vulnerability and of a danger or a threat.
Rationale	An argument that provides the justification for the selection of an engineering element.
	Identifier, name, description (rationale, reasons for defining a verification action, a verification procedure, for using a verification tool, etc.)

Methods and Techniques

There are several verification techniques to check that an element or a system conforms to its design references, or its specified requirements. These techniques are almost the same as those used for validation, though the application of the techniques may differ slightly. In particular, the purposes are different; verification is used to detect faults/defects, whereas validation is used to provide evidence for the satisfaction of (system and/or stakeholder) requirements. Table 3 below provides descriptions of some techniques for verification.

Table 3. Verification Techniques. (SEBoK Original)

Verification Technique	Description
Inspection	Technique based on visual or dimensional examination of an element; the verification relies on the human senses or uses simple methods of measurement and handling. Inspection is generally non-destructive, and typically includes the use of sight, hearing, smell, touch, and taste, simple physical manipulation, mechanical and electrical gauging, and measurement. No stimuli (tests) are necessary. The technique is used to check properties or characteristics best determined by observation (e.g. - paint color, weight, documentation, listing of code, etc.).
Analysis	Technique based on analytical evidence obtained without any intervention on the submitted element using mathematical or probabilistic calculation, logical reasoning (including the theory of predicates), modeling and/or simulation under defined conditions to show theoretical compliance. Mainly used where testing to realistic conditions cannot be achieved or is not cost-effective.
Analogy or Similarity	Technique based on evidence of similar elements to the submitted element or on experience feedback. It is absolutely necessary to show by prediction that the context is invariant that the outcomes are transposable (models, investigations, experience feedback, etc.). Similarity can only be used if the submitted element is similar in design, manufacture, and use; equivalent or more stringent verification actions were used for the similar element, and the intended operational environment is identical to or less rigorous than the similar element.
Demonstration	Technique used to demonstrate correct operation of the submitted element against operational and observable characteristics without using physical measurements (no or minimal instrumentation or test equipment). Demonstration is sometimes called 'field testing'. It generally consists of a set of tests selected by the supplier to show that the element response to stimuli is suitable or to show that operators can perform their assigned tasks when using the element. Observations are made and compared with predetermined/expected responses. Demonstration may be appropriate when requirements or specification are given in statistical terms (e.g. meant time to repair, average power consumption, etc.).
Test	Technique performed onto the submitted element by which functional, measurable characteristics, operability, supportability, or performance capability is quantitatively verified when subjected to controlled conditions that are real or simulated. Testing often uses special test equipment or instrumentation to obtain accurate quantitative data to be analyzed.
Sampling	Technique based on verification of characteristics using samples. The number, tolerance, and other characteristics must be specified to be in agreement with the experience feedback.

Practical Considerations

Key pitfalls and good practices related to this topic are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing System Verification are provided in Table 4.

Table 4. Major Pitfalls with System Verification (SEBoK Original)

Pitfall	Description
Confusion between verification and validation	Confusion between verification and validation causes developers to take the wrong reference/baseline to define verification and validation actions and/or to address the wrong level of granularity (detail level for verification, global level for validation).
No verification strategy	One overlooks verification actions because it is impossible to check every characteristic or property of all system elements and of the system in any combination of operational conditions and scenarios. A strategy (justified selection of verification actions against risks) has to be established.
Save or spend time	Skip verification activity to save time.
Use only testing	Use only testing as a verification technique. Testing requires checking products and services only when they are implemented. Consider other techniques earlier during design; analysis and inspections are cost effective and allow discovering early potential errors, faults, or failures.
Stop verifications when funding is diminished	Stopping the performance of verification actions when budget and/or time are consumed. Prefer using criteria such as coverage rates to end verification activity.

Proven Practices

Some proven practices gathered from the references are provided in Table 5.

Table 5. Proven Practices with System Verification. (SEBoK Original)

Practice	Description
Start verifications early in the development	The more the characteristics of an element are verified early in the project, the more the corrections are easy to do and less the error will have consequences on schedule and costs.
Define criteria ending verifications	Carrying out verification actions without limits generates a risk of drift for costs and deadlines. Modifying and verifying in a non-stop cycle until a get a perfect system is the best way to never supply the system. Thus, it is necessary to set limits of cost, time, and a maximum number of modification loops back for each verification action type, ending criteria (percentages of success, error count detected, coverage rate obtained, etc.).
Involve design responsible with verification	Include the verification responsible in the designer team or include some designer onto the verification team.

References

Works Cited

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods.* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

INCOSE. 2012. *INCOSE Systems Engineering Handbook,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Lake, J. 1999. "V & V in Plain English." International Council on Systems Engineering (NCOSE) 9th Annual International Symposium, Brighton, UK, 6-10 June 1999.

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

Primary References

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

Additional References

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

ECSS. 2009. *Systems Engineering General Requirements.* Noordwijk, Netherlands: Requirements and Standards Division, European Cooperation for Space Standardization (ECSS), 6 March 2009. ECSS-E-ST-10C.

MITRE. 2011. "Verification and Validation." in *Systems Engineering Guide.* Accessed 11 March 2012 at [[1]].

SAE International. 1996. *Certification Considerations for Highly-Integrated or Complex Aircraft Systems.* Warrendale, PA, USA: SAE International, ARP475.

SEI. 2007. "Measurement and Analysis Process Area" in *Capability Maturity Model Integrated (CMMI) for Development, version 1.2.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/test_evaluation/verification_validation.html

System Validation

System Validation is a set of actions used to check the compliance of any element (a system element, a system, a document, a service, a task, a system requirement, etc.) with its purpose and functions. These actions are planned and carried out throughout the life cycle of the system. Validation is a generic term that needs to be instantiated within the context it occurs. When understood as a process, validation is a transverse activity to every life cycle stage of the system. In particular, during the development cycle of the system, the validation process is performed in parallel with the system definition and system realization processes and applies to any activity and product resulting from this activity. The validation process is not limited to a phase at the end of system development, but generally occurs at the end of a set of life cycle tasks or activities, and always at the end of each milestone of a development project. It may be performed on an iterative basis on every produced engineering element during development and may begin with the validation of the expressed stakeholder requirements. When the validation process is applied to the system when completely integrated, it is often called *final validation*. It is important to remember that while system validation is separate from verification, the activities are complementary and intended to be performed in conjunction.

Definition and Purpose

Validation is the confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled. With a note added in ISO 9000:2005: *validation is the set of activities that ensure and provide confidence that a system is able to accomplish its intended use, goals, and objectives (i.e., meet stakeholder requirements) in the intended operational environment* (ISO 2005).

The purpose of validation, as a generic action, is to establish the compliance of any activity output as compared to inputs of the activity. It is used to provide information and evidence that the transformation of inputs produced the expected and *right* result. Validation is based on tangible evidence; i.e., it is based on information whose veracity can be demonstrated by factual results obtained from techniques or methods such as inspection, measurement, test, analysis, calculation, etc. Thus, to validate a system (product, service, or enterprise) consists of demonstrating that it satisfies its system requirements and eventually the stakeholder's requirements depending on contractual practices. From a global standpoint, the purpose of validating a system is to acquire confidence in the system's ability to achieve its intended mission, or use, under specific operational conditions.

Principles

Concept of Validation Action

Why Validate?

The primary goal of systems engineering (SE) is to develop a solution that meets the needs and requirements of stakeholders. Validation is the process by which engineers ensure that the system will meet these needs and requirements.

A **validation action** is defined and then performed (see Figure 1, below).

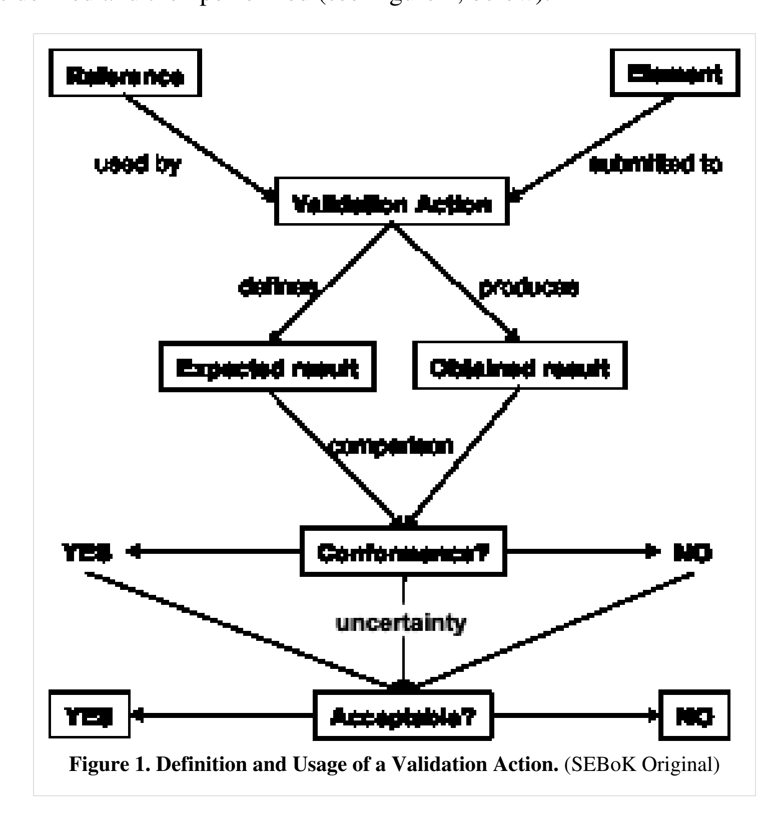


Figure 1. Definition and Usage of a Validation Action. (SEBoK Original)

A validation action applied to an engineering element includes the following:

- Identification of the element on which the validation action will be performed.
- Identification of the reference that defines the expected result of the validation action.

Performing the validation action includes the following:

- Obtaining a result by performing the validation action onto the submitted element.
- Comparing the obtained result with the expected result.
- Deducing the degree of compliance of the element.
- Deciding on the acceptability of this compliance, because sometimes the result of the comparison may require a value judgment to decide whether or not to accept the obtained result as compared to the relevance of the context of use.

Note: If there is uncertainty about compliance, the cause could come from ambiguity in the requirements.

What to Validate?

Any engineering element can be validated using a specific reference for comparison, such as stakeholder requirements, system requirements, functions, system elements, documents, etc. Examples are provided in Table 1 below:

Table 1. Examples of Validated Items (SEBoK Original)

Items	Explanation for Validation
Document	To validate a document is to make sure its content is compliant with the inputs of the task that produced the document.
Stakeholder Requirement and System Requirement	To validate a stakeholder requirement its make sure its content is justified and relevant to stakeholders' expectations, complete and expressed in the language of the customer or end user. To validate a system requirement is to make sure its content translates correctly and/or accurately a stakeholder requirement in the language of the supplier.
Design	To validate the design of a system (logical and physical architectures) is to demonstrate that it satisfies its system requirements.
System	To validate a system (product, service, or enterprise) is to demonstrate that the product, service, or enterprise satisfies its system requirements and/or its stakeholder requirements.
Activity	To validate an activity or a task is to make sure its outputs are compliant with its inputs.
Process	To validate a process is to make sure its outcomes are compliant with its purpose.

Validation versus Verification

The Verification versus Validation section of the System Verification article gives fundamental differences between the two concepts and associated processes. The Table 2 provides information to help understand these differences.

Table 2. Verification and Validation Differences (may vary with context). (SEBoK Original)

Point of View	Verification	Validation
Purpose of the Activity	Detect, identify faults/defects (supplier oriented)	Acquire confidence (end user oriented)
Idea behind the Term	Based on truth (objective/unbiased)	Vased on value judgement (more subjective)
Level of Concern	Detail and local	Global in the context of use
Vision	Glass box (how it runs inside)	Black box (application of inputs provides the expected effect)
Basic Method	Fine-tooth comb	Traceability matrix
System (Product, Service, Enterprise)	"Done Right" (respects the state of the art); focus on (physical) characteristics	"Does Right" (produces the expected effect); focus on services, functions
Baseline Reference for Comparison (Product, Service, Enterprise)	System design	System requirements (and stakeholder requirements)
Order of Performance	First	Second
Organization of Activity	Verification actions are defined and/or performed by development/designer team	Validation actions are defined and/or performed by experts and external members to development/designer team

Validation, Final Validation, and Operational Validation

System validation concerns the global system seen as a whole and is based on the totality of requirements (system requirements, stakeholders requirements, etc.), but it is obtained gradually throughout the development stage in three non-exclusive ways:

- accumulating the results of verification actions and validation actions provided by the application of corresponding processes to every engineering element;
- performing final validation actions to the complete, integrated system in an industrial environment (as close as possible to the operational environment); and
- performing operational validation actions on the complete system in its operational environment (context of use).

Verification and Validation Level per Level

It is impossible to carry out only a single global validation on a complete, integrated complex system. The sources of faults/defects could be important and it would be impossible to determine the causes of non-conformance manifested during this global check. Generally, the system-of-interest (SoI) has been decomposed during design in a set of layers of systems. Thus, every system and system element is verified, validated, and possibly corrected before being integrated into the parent system of the higher level, as shown in Figure 2.

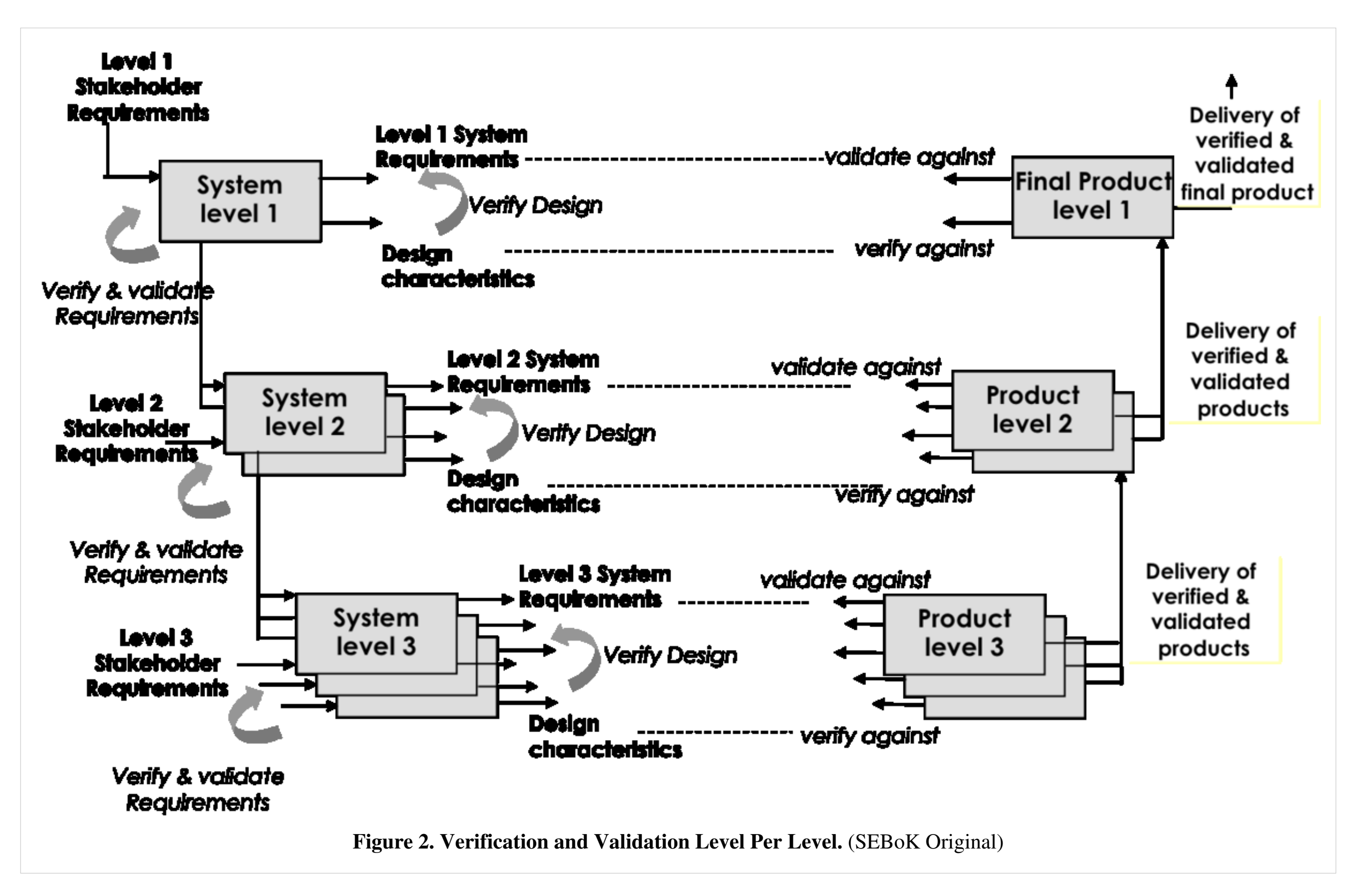


Figure 2. Verification and Validation Level Per Level. (SEBoK Original)

As necessary, systems and system elements are partially integrated in subsets in order to limit the number of properties to be verified within a single step. For each level, it is necessary to perform a set of final validation actions to ensure that features stated at preceding levels are not damaged. Moreover, a compliant result obtained in a given environment can turn into a non-compliant result if the environment changes. Thus, as long as the system is not completely integrated and/or doesn't operate in the real operational environment, no result should be regarded as definitive.

Verification Actions and Validation Actions Inside and Transverse to Levels

Inside each level of system decomposition, verification actions and validation actions are performed during system definition and system realization. This is represented in Figure 3 for the upper levels, and in Figure 4 for the lower levels. Stakeholder requirements definition and operational validation make the link between the two levels of the system decomposition.

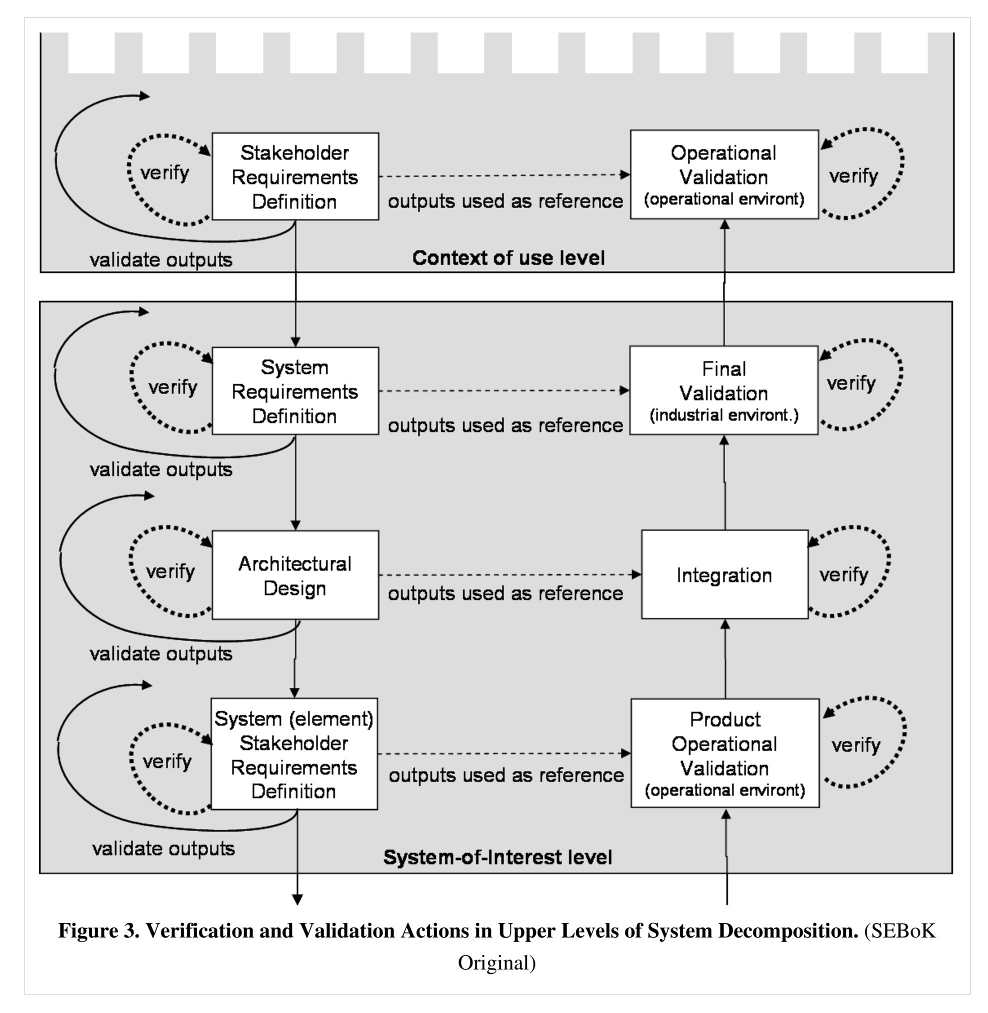


Figure 3. Verification and Validation Actions in Upper Levels of System Decomposition. (SEBoK Original)

Operational validation of system element requirements and products makes the link between the two lower levels of the decomposition. See Figure 4 below.

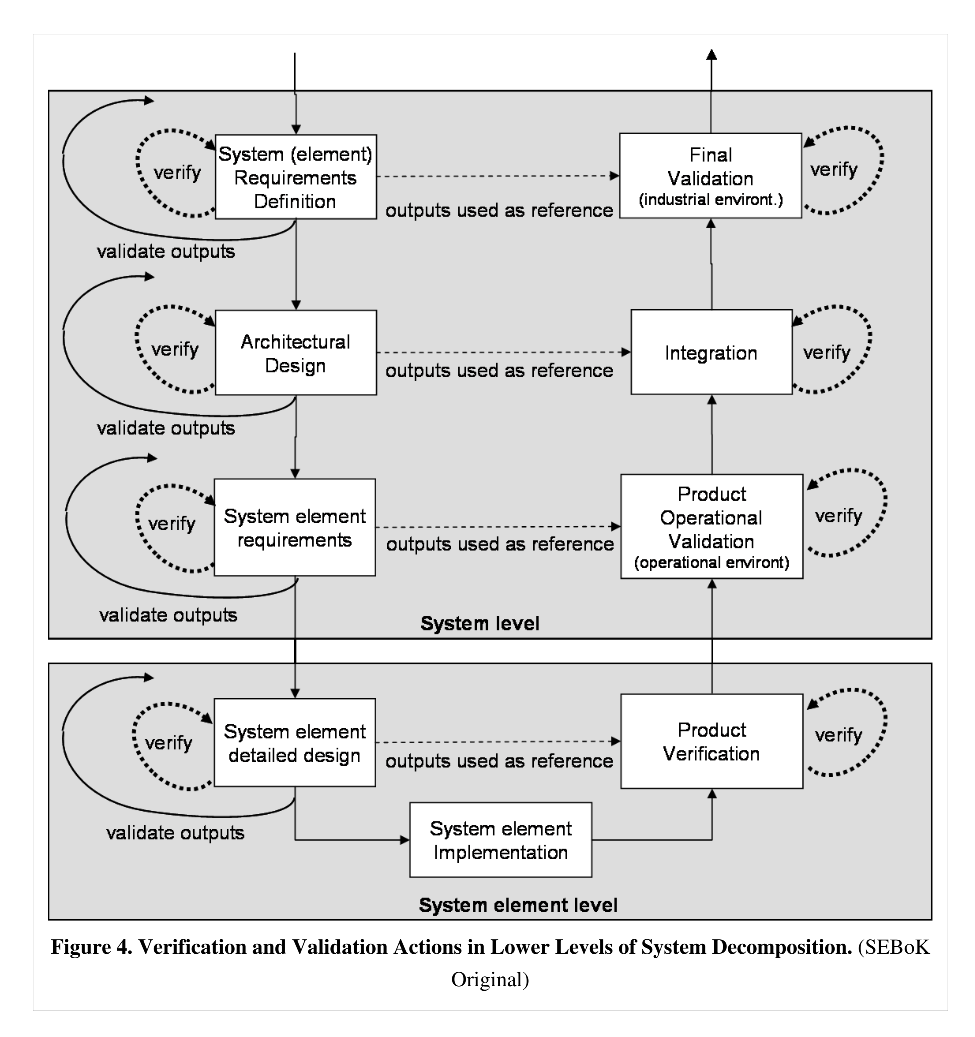


Figure 4. Verification and Validation Actions in Lower Levels of System Decomposition. (SEBoK Original)

Note: The last level of system decomposition is dedicated to the realization of system elements and the vocabulary and number of activities may be different from what is seen in Figure 4.

Verification and Validation Strategy

The difference between verification and validation is especially useful for elaborating on the integration strategy, the verification strategy, and the validation strategy. In fact, the efficiency of system realization is gained by optimizing the three strategies together to form what is often called the verification and validation strategy. This optimization consists of defining and performing the minimum number of verification and validation actions but detecting the maximum number of errors/faults/defects and achieving the maximum level of confidence in the system. The optimization takes into account the risks potentially generated if some verification actions or validation actions are excluded.

Process Approach

Purpose and Principles of the Approach

The purpose of the validation process is to provide objective evidence that the services provided by a system in use comply with stakeholder requirements and achieve its intended use in its intended operational environment (ISO/IEC/IEEE 15288 2015). The validation process performs a comparative assessment and confirms that the stakeholder requirements are correctly defined. Where variance is identified, it is recorded to guide future corrective actions. System validation is ratified by stakeholders (ISO/IEC/IEEE 15288 2015).

The validation process demonstrates that the realized end product satisfies its stakeholders' (customers' or other interested parties') expectations within the intended operational environments with validation performed by anticipated operators and/or users (NASA 2007, 1-360). Each system element, system, and the complete SoI are compared against their own applicable requirements (system requirements and stakeholder requirements). This means that the validation process is instantiated as many times as necessary during the global development of the system.

In order to ensure that validation is feasible, the implementation of requirements must be verifiable onto a defined element. It is essential to ensure that requirements are properly written, i.e., quantifiable, measurable, unambiguous, etc.. In addition, verification/validation requirements are often written in conjunction with stakeholder and system requirements and provide a method for demonstrating the implementation of each system requirement or stakeholder requirement.

Generic inputs are references of requirements applicable to the submitted element. If the element is a system, inputs are system requirements and stakeholder requirements.

Generic outputs are the validation plan that includes validation strategy, selected validation actions, validation procedures, validation tools, validated elements or systems, validation reports, issue/trouble reports, and change requests on requirements or on the system.

Activities of the Process

Major activities and tasks performed during this process include the following:

- Establish a validation strategy (often drafted in a validation plan). This activity is carried out concurrently to system definition activities:
  - Identify the validation scope that is represented by (system and/or stakeholder) requirements; normally, every requirement should be checked as the number of validation actions can be high.
  - Identify constraints according to their origin (technical feasibility, management constraints as cost, time, availability of validation means or qualified personnel, and contractual constraints that are critical to the mission) that limit or increase potential validation actions.
  - Define appropriate verification/validation techniques to be applied, such as inspection, analysis, simulation, review, testing, etc., depending on the best step of the project to perform every validation action according to constraints.
  - Consider a trade-off of what should be validated (scope) while taking into account all constraints or limits and deduce what can be validated objectively; selection of validation actions would be made according to the type of system, objectives of the project, acceptable risks, and constraints.
  - Optimize the validation strategy to define the most appropriate validation technique for every validation action, define necessary validation means (tools, test-benches, personnel, location, and facilities) according to the selected validation technique, schedule the execution of validation actions in the project steps or milestones, and define the configuration of elements submitted to validation actions (this is primarily about testing on physical elements).
- Perform validation actions, including the following tasks:
  - Detail each validation action, in particular, note the expected results, the validation technique to be applied, and the corresponding means necessary (equipment, resources, and qualified personnel).
  - Acquire validation means used during the system definition steps (qualified personnel, modeling tools, mocks-up, simulators, and facilities), then those means used during integration and final and operational steps (qualified personnel, validation tools, measuring equipment, facilities, validation procedures, etc.).
  - Carry out validation procedures at the right time, in the expected environment, with the expected means, tools, and techniques.

- Capture and record results obtained when performing validation actions using validation procedures and means.
- Analyze the obtained results and compare them to the expected results. Decide if they comply acceptably. Record whether the decision and status are compliant or not, and generate validation reports and potential issue/trouble reports, as well as change requests on (system or stakeholder) requirements as necessary.
- Control the process using following tasks:
  - Update the validation plan according to the progress of the project; in particular, planned validation actions can be redefined because of unexpected events.
  - Coordinate validation activities with the project manager regarding the schedule, acquisition of means, personnel, and resources. Coordinate with the designers for issue/trouble/non-conformance reports. Coordinate with the configuration manager for versions of physical elements, design baselines, etc.

Artifacts and Ontology Elements

This process may create several artifacts, such as:

- a validation plan (contains the validation strategy)
- a validation matrix (contains for each validation action, submitted element, applied technique, step of execution, system block concerned, expected result, obtained result, etc.)
- validation procedures (describe the validation actions to be performed, the validation tools needed, the validation configuration, resources, personnel, schedule, etc.)
- validation reports
- validation tools
- the validated element
- issue, non-conformance, and trouble reports
- change requests on requirements, products, services, and enterprises

This process utilizes the ontology elements of Table 3.

Table 3. Main Ontology Elements as Handled within Validation. (SEBoK Original)

Element	Definition
	Attributes (examples)
Validation Action	A validation action describes what must be validated (the element as reference), on which element, the expected result, the verification technique to apply, on which level of decomposition.
	Identifier, name, description
Validation Procedure	A validation procedure groups a set of validation actions performed together (as a scenario of tests) in a given validation configuration.
	Identifier, name, description, duration, unit of time
Validation Tool	A validation tool is a device or physical tool used to perform validation procedures (test bench, simulator, cap/stub, launcher, etc.).
	Identifier, name, description
Validation Configuration	A validation configuration groups the physical elements necessary to perform a validation procedure.
	Identifier, name, description

Risk	An event having a probability of occurrence and a gravity degree on its consequence onto the system mission or on other characteristics (used for technical risk engineering).
	Identifier, name, description, status
Rationale	An argument that provides the justification for the selection of an engineering element.
	Identifier, name, description (rationale, reasons for defining a validation action, a validation procedure, for using a validation tool, etc.)

Methods and Techniques

The validation techniques are the same as those used for verification, but their purposes are different; verification is used to detect faults/defects, whereas validation is used to prove the satisfaction of (system and/or stakeholder) requirements.

The **validation traceability matrix** is introduced in the stakeholder requirements definition topic. It may also be extended and used to record data, such as a validation actions list, selected validation techniques to validate implementation of every engineering element (in particular stakeholder and system requirements), expected results, and obtained results when validation actions have been performed. The use of such a matrix enables the development team to ensure that selected stakeholder and system requirements have been checked, or to evaluate the percentage of validation actions completed.

Practical Considerations

Key pitfalls and good practices related to system validation are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing system validation are provided in Table 4.

Table 4. Major Pitfalls with System Validation. (SEBoK Original)

Pitfall	Description
Start validation at the end of the project	A common mistake is to wait until the system has been entirely integrated and tested (design is qualified) to perform any sort of validation. Validation should occur as early as possible in the [product] life cycle (Martin 1997).
Use only testing	Use only testing as a validation technique. Testing requires checking products and services only when they are implemented. Consider other techniques earlier during design; analysis and inspections are cost effective and allow discovering early potential errors, faults, or failures.
Stop validation when funding is diminished	Stop the performance of validation actions when budget and/or time are consumed. Prefer using criteria such as coverage rates to end validation activity.

Proven Practices

Some good practices gathered from the references are provided in Table 5.

Table 5. Proven Practices with System Validation. (SEBoK Original)

Practice	Description
Start Validation Plan Early	It is recommended to start the drafting of the validation plan as soon as the first requirements applicable to the system are known. If the writer of the requirements immediately puts the question to know how to validate whether the future system will answer the requirements, it is possible to: • detect the unverifiable requirements • anticipate, estimate cost, and start the design of validation means (as needed) such as test-benches, simulators • avoid cost overruns and schedule slippages
Verifiable Requirements	According to Buede, a requirement is verifiable if a "finite, cost-effective process has been defined to check that the requirement has been attained." (Buede 2009) Generally, this means that each requirement should be quantitative, measurable, unambiguous, understandable, and testable. It is generally much easier and more cost-effective to ensure that requirements meet these criteria while they are being written. Requirement adjustments made after implementation and/or integration are generally much more costly and may have wide-reaching redesign implications. There are several resources which provide guidance on creating appropriate requirements - see the system definition knowledge area, stakeholder requirements, and system requirements topics for additional information.
Document Validation Actions	It is important to document both the validation actions performed and the results obtained. This provides accountability regarding the extent to which system, system elements, subsystems fulfill system requirements and stakeholders' requirements. These data can be used to investigate why the system, system elements, subsystems do not match the requirements and to detect potential faults/defects. When requirements are met, these data may be reported to organization parties. For example, in a safety critical system, it may be necessary to report the results of safety demonstration to a certification organization. Validation results may be reported to the acquirer for contractual aspects or to internal company for business purpose.
Involve Users with Validation	Validation will often involve going back directly to the users to have them perform some sort of acceptance test under their own local conditions.
Involve	Often the end users and other relevant stakeholders are involved in the validation process.

The following are elements that should be considered when practicing any of the activities discussed as a part of system realization:

- Confusing verification and validation is a common issue. Validation demonstrates that the product, service, and/or enterprise as provided, fulfills its intended use, whereas verification addresses whether a local work product properly reflects its specified requirements. Validation actions use the same techniques as the verification actions (e.g., test, analysis, inspection, demonstration, or simulation).
- State who the witnesses will be (for the purpose of collecting the evidence of success), what general steps will be followed, and what special resources are needed, such as instrumentation, special test equipment or facilities, simulators, specific data gathering, or rigorous analysis of demonstration results.
- Identify the test facility, test equipment, any unique resource needs and environmental conditions, required qualifications and test personnel, general steps that will be followed, specific data to be collected, criteria for repeatability of collected data, and methods for analyzing the results.

References

Works Cited

Buede, D. M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ: John Wiley & Sons Inc.

INCOSE. 2012. *INCOSE Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Primary References

INCOSE. 2012. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC 15288:2015.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Additional References

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ: John Wiley & Sons Inc.

DAU. February 19, 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense.

ECSS. 2009. Systems engineering general requirements. Noordwijk, Netherlands: Requirements and Standards Division, European Cooperation for Space Standardization (ECSS), ECSS-E-ST-10C. 6 March 2009.

MITRE. 2011. "Verification and Validation." *Systems Engineering Guide.* Accessed 11 March 2012 at [[1]].

SAE International. 1996. *Certification considerations for highly-integrated or complex aircraft systems.* Warrendale, PA, USA: SAE International, ARP475

SEI. 2007. *Capability maturity model integrated (CMMI) for development*, version 1.2, measurement and analysis process area. Pittsburg, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: System Deployment and Use

System Deployment and Use

System deployment and use are critical systems engineering (SE) activities that ensure that the developed system is operationally acceptable and that the responsibility for the effective, efficient, and safe operations of the system is transferred to the owner. Considerations for deployment and use must be included throughout the system life cycle.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Deployment
- Operation of the System
- System Maintenance
- Logistics

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

Overview

System deployment involves the transition of the capability to the ultimate end-user, as well as transition of support and maintenance responsibilities to the post-deployment support organization or organizations. It may include a period of reliability demonstration tests and the phasing out of legacy systems that the developed system replaces.

System use includes a continual assessment of the operational effectiveness of the deployed system or service, identification of mission threat and operational risk, and performance of the actions required to maintain operational effectiveness or evolve the capability to meet changing needs. Evolution of the operational system may occur with smaller maintenance actions or, if the changes cross an agreed-to threshold (complexity, risk, cost, etc.), may require a formal development project with deliberate planning and SE activities resulting in an enhanced system. As the operational phase is generally the longest in the system life cycle, activities that may occur during operation are allocated between two knowledge areas (KAs): System Deployment and Use and Product and Service Life Management.

The Product and Service Life Management knowledge area (KA) specifically deals with SE activities required for system evolution and end of system life including service life extension (SLE), capability updates, upgrades, and modernization during system operation, and system disposal and retirement. In contrast, the System Deployment and Use KA specifically deals with activities required to ensure that system operation can continue as expected. Planning for system deployment and use should begin early in the SE process to ensure successful transition into operational use.

System Deployment and Use Fundamentals

System deployment and use includes the processes used to plan for and manage the transition of new or evolved systems and capabilities into operational use and the transition of support responsibilities to the eventual maintenance or support organization. The *use* stage normally represents the longest period of a system life cycle and, hence, generally accounts for the largest portion of the life cycle cost. These activities need to be properly managed in order to evaluate the actual system performance, effectiveness, and cost in its intended environment and within its specified utilization over its life cycle. Included in use fundamentals are the aspects of continuation of personnel training and certification.

As part of deployment/transition activities special conditions that may apply during the eventual decommissioning or disposal of the system are identified and accommodated in life cycle plans and system architectures and designs (See the System Definition KA for additional information). SE leadership ensures the developed system meets specified requirements, that it be used in the intended environment, and that when the system is transitioned into operation, it achieves the users' defined mission capabilities and can be maintained throughout the intended life cycle.

SE ensures that plans and clear criteria for transition into operation are developed and agreed to by relevant stakeholders and that planning is completed for system maintenance and support after the system is deployed. These plans should generally include reasonable accommodation for planned and potential evolution of the system and its eventual removal from operational use (for additional information on evolution and retirement, please see the Product and Service Life Management KA).

References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Deployment

As part of system deployment, on-site installation, check-out, integration, and testing must be carried out to ensure that the system is fit to be deployed into the field and/or put into an operational context. *Transfer* is the process that bridges the gap between qualification and use; it deals explicitly with the handoff from development to logistics, operations, maintenance, and support.

Definition & Purpose

There are many different approaches to transition, or deployment, and many different views on what is included within transition. The SEBoK uses the ISO/IEC/IEEE 15288 definition of transition, as seen below (ISO/IEC/IEEE 15288 2015):

> *[The transition] process installs a verified system, together with relevant enabling systems, e.g., operating system, support system, operator training system, user training system, as defined in agreements. This process is used at each level in the system structure and in each stage to complete the criteria established for exiting the stage.*

Thinking in a linear fashion, the system is transitioned into operation and then would be used and maintained in the operational environment. However, there are other views on transition. For example, the NASA *Systems Engineering Handbook* states that transition can include delivery for end-use as well as delivery of components for integration (NASA 2007). Using this view, transition is the mechanism for moving system components from implementation activities into integration activities. The NASA discussion of transition also implies that transition can include sustainment activities:

> *The act of delivery or moving of a product from the location where the product has been implemented or integrated, as well as verified and validated, to a customer.*

Many systems are deployed using an iterative or evolutionary approach where operationally useful capabilities are developed and deployed incrementally. While these operationally useful capabilities are fully deployed and transitioned into operational use, transition of logistics, maintenance, and support may occur incrementally or be delayed until after the full system capability is delivered.

Process Approaches

Just as there are multiple views on the definition of transition and deployment, there are also several ways to divide the activities required for transition. For example, the NASA *Systems Engineering Handbook* definition of transition states: *This act can include packaging, handling, storing, moving, transporting, installing, and sustainment activities* (2007). However, the SEBoK includes the topic of *sustainment* as separate from transition; this is instead covered under the maintenance and logistics topics. The International Council on Systems Engineering (INCOSE) views the transition process as two-step: planning and performance. Though there are several processes for deployment and transition, most generally include the following activities:

- **Develop a Deployment/Transition Strategy** - Planning for transition activities would ideally begin early in the SE life cycle, though it is possible to conduct these activities concurrently with realization activities. Planning should generally include some consideration of the common lower-level activities of installation, checkout, integration, and testing. Such activities are crucial to demonstrate that the system and the interfaces with the operational environment can function as intended and meet the contractual system specifications. For these activities to be effectively managed and efficiently implemented, the criteria, responsibility, and procedures for carrying out these activities should be clearly established and agreed upon during the planning phase.
- **Develop Plans for Transitioning Systems** - or system capabilities into operational use and support. Transition plans for the system or incremental system capabilities should be consistent with the overall transition strategy

and agreed to by relevant stakeholders. Planning for transition will often include establishing a strategy for support, which may include organic support infrastructures, contractor logistics support, or other sources (Bernard et al. 2005, 1-49). It can also include defining the levels of support to be established. The strategy is important because it drives most of the other transition planning activities, as well as product design considerations. Transition plans should include considerations for coordination with the following activities:

- **Installation** - Installation generally refers to the activities required to physically instate the system; this will likely include connecting interfaces to other systems such as electrical, computer, or security systems, and may include software interfaces as well. Installation planning should generally document the complexity of the system, the range of environmental conditions expected in the operational environment, any interface specifications, and human factors requirements such as safety. When real-world conditions require changes in the installation requirements, these should be documented and discussed with the relevant stakeholders.
- **Integration** - Though system integration activities will generally be performed prior to installation, there may be additional steps for integrating the system into its operational setting. Additionally, if the system is being delivered incrementally, there will likely be integration steps associated with the transition (for more information on integration, please see the System Realization knowledge area (KA)).
- **Verification and Validation (V&V)** - At this stage, V&V for physical, electrical, and mechanical checks may be performed in order to verify that the system has been appropriately installed. Acceptance tests conducted after delivery may become part of this process (for additional information on V&V, please see the System Realization KA). There are several types of acceptance tests which may be used:
- **On-site Acceptance Test (OSAT)** - This test includes any field acceptance testing and is performed only after the system has successfully been situated in the operational environment. It may consist of functional tests to demonstrate that the system is functioning and performing properly.
  - *Field Acceptance Test* - This test includes flight and sea acceptance tests; it is performed, if applicable, only after the system has successfully passed the OSAT. The purpose of field testing is to demonstrate that the system meets the performance specifications called for in the system specifications in the actual operating environment.
  - *Operational Test and Evaluation (OT&E)* - An OT&E consists of a test series designed to estimate the operational effectiveness of the system.
  - *Evaluate the readiness of the system to transition into operations* - This is based upon the transition criteria identified in the transition plan. These criteria should support an objective evaluation of the system's readiness for transition. The integration, verification, and validation activities associated with transition may be used to gauge whether the system meets transition criteria.
  - *Analyze the results of transition activities throughout and any necessary actions* - As a result of analysis, additional transition activities and actions may be required. The analysis may also identify areas for improvement in future transition activities.

Some common issues that require additional considerations and SE activities are the utilization or replacement of legacy systems. It is also common for an organization to continue testing into the early operational phase. The following activities support these circumstances:

- **System Run-in** - After the successful completion of the various acceptance tests, the system(s) will be handed over to the user or designated post-deployment support organization. The tested system(s) may have to be verified for a stated period (called the system run-in, normally for one to two years) for the adequacy of reliability and maintainability (R&M) and integrated logistics support (ILS) deliverables. R&M are vital system operational characteristics having a dominant impact upon the operational effectiveness, the economy of in-service maintenance support, and the life cycle cost (LCC).
- **Phasing-In/Phasing-Out** - The need for phasing-in will usually be identified during the system definition, when it is clear that the new system entails the replacement of an existing system(s) (for additional information, please

see the System Definition KA). These activities should help to minimize disruption to operations and, at the same time, minimize the adverse effect on operational readiness. It is also important that the phasing-in of a new system and the phasing-out of an existing system occur in parallel with the systems activities of the system run-in to maximize resource utilization. Other aspects of phasing-in/phasing-out to be considered include:

- Proper planning for the phasing out of an existing system (if necessary).
- For multi-user or complex systems, phase-by-phase introduction of the system according to levels of command, formation hierarchy, etc.
- Minimum disruption to the current operations of the users.
- Establishment of a feedback system from users on problems encountered in operation, etc.
- Disposal process including handling of hazardous items, cost of disposal, approval etc.

Applicable Methods & Tools

A system may have to undergo reliability demonstration testing (RDT) to ensure that it meets its contractual R&M guarantees. RDT is conducted under actual field conditions, especially for large systems purchased in small quantity. During RDT, the system is operated in the field within stated test duration and all field data are systematically recorded. At the end of the test period, analysis of the RDT data is performed. Data analysis should facilitate determination of system reliability. One possible output of this analysis is shown in Figure 1 below.

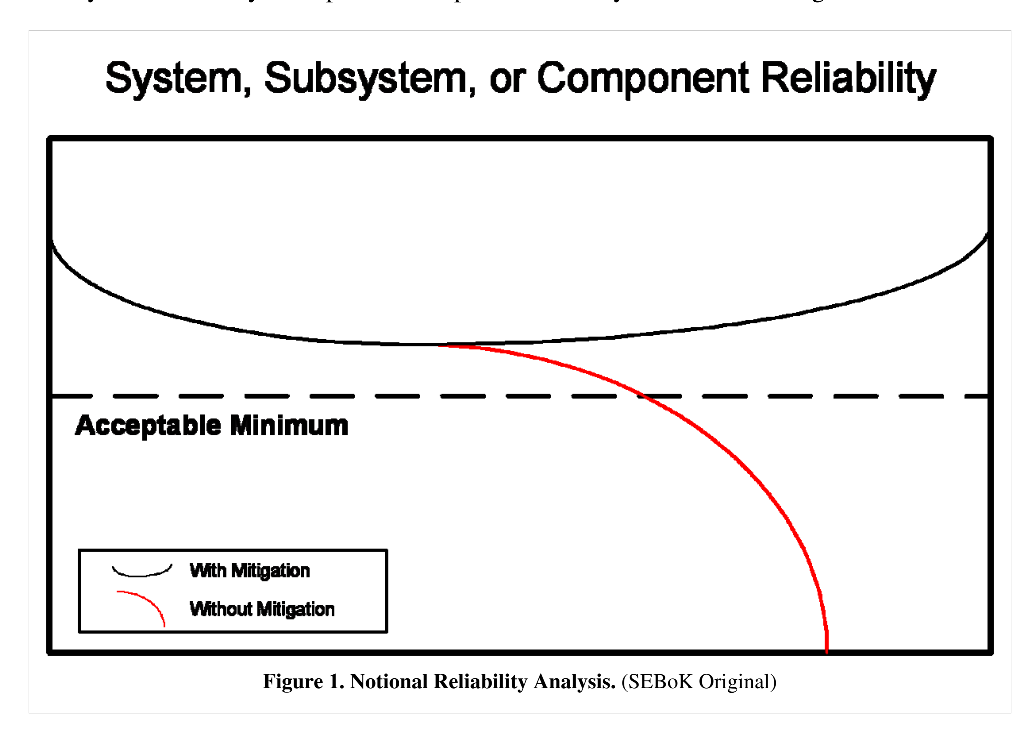


Figure 1. Notional Reliability Analysis. (SEBoK Original)

References

Works Cited

Bernard, S., B. Gallagher, R. Bate, H. Wilson. 2005. *CMMI® Acquisition Module (CMMI-AM),* version 1.1. Pittsburg, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU). CMU/SEI-2005-TR-011.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Primary References

INCOSE. 2011. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Operation of the System

The role of systems engineering (SE) during the operation of a system consists of ensuring that the system maintains key mission and business functions and is operationally effective. The systems engineer is one of the stakeholders who ensures that maintenance actions and other major changes are performed according to the long-term vision of the system. Both the maintenance actions and any implemented changes must meet the evolving needs of owning and operating stakeholders consistent with the documented and approved architecture. SE considerations will also include the eventual decommissioning or disposal of the system so that the disposal occurs according to disposal/retirement plans. Those plans must take into account and be compliant with relevant laws and regulations (for additional information on disposal or retirement, please see the Product and Service Life Management knowledge area (KA)). When the system-of-interest (SoI) replaces an existing or legacy system, it may be necessary to manage the migration between systems such that stakeholders do not experience a breakdown in services (INCOSE 2012).

Definition & Purpose

This process assigns personnel to operate the system and monitors the services and operator-system performance. In order to sustain services, it identifies and analyzes operational problems in relation to agreements, stakeholder requirements, and organizational constraints (ISO/IEC/IEEE 2015).

The concept of operations (ConOps) establishes the foundation for initial design specifications according to the long-term vision. It is also possible that pre-planned program improvements (P3I) had been generated based on expected evolving requirements. Throughout the systems life cycle the operation of the system requires the systems engineer to be an active participant in reviews, change management and integrated master schedule activities to ensure the system operations continue to meet the evolving needs of stakeholders, and are consistent with the architecture through the eventual decommissioning or disposal of the system. In the event of decommissioning, a systems engineer must ensure disposal/retirement plans are compliant with relevant laws and regulations (for additional information on disposal or retirement, see the Product and Service Life Management KA).

Two additional areas are of interest to the systems engineer during system operation require special attention. First, it may be determined that a system is at the end of its life cycle, but the cost of replacing the system with a completely new design is too expensive. In this case, there will be intense engineering activities for service life extension program (SLEP). The SLEP solution will take into account obsolescence issues, diminishing manufacturing sources and material shortages (DMSMS), and changes in ConOps. Secondly, in the event that a new SoI is designed and produced as a complete replacement for an existing or legacy system, it will be necessary to manage the migration between systems such that stakeholders do not experience a breakdown in services (INCOSE 2012).

Process Approaches

During the operational phase, SE activities ensure the system maintains certain operational attributes and usefulness throughout its expected life span. Maintaining operational effectiveness consists of evaluating certain operationally relevant attributes and trends, taking actions to prevent degradation of performance, evolving the system to meet changing mission or business needs (see the Product and Service Life Management KA), and eventually decommissioning the system and disposing of its components. During operation, data would be collected to evaluate the system and determine if changes should be made. It is important to include the process for data collection during operations when considering design and ConOps. In some cases, data may be collected by sensors and reported autonomously. In other cases, operators will identify and report on performance during operations. The systems engineer needs to understand how all data will be collected and presented for further analysis. The systems engineer will be involved in analysis of this data in several areas, including the following:

- Updating training and development of new training as required for operational and support personnel. Training is generally developed early with system design and production and executed during integration and operations. Determination of training updates or changes will be based on evaluation of the operational and support personnel.
- Evaluation of operational effectiveness. Early in the planning phases of a new system or capability, measures of operational effectiveness are established based on mission and business goals. These measures are important during system operation. These attributes are unique for each system and represent characteristics describing the usefulness of the system as defined and agreed to by system stakeholders. Systems engineers monitor and analyze these measurements and recommend actions.
- Failure reporting and corrective actions (FRACA) activities will involve the collection and analysis of data during operations. FRACA data will provide trends involving failures that may require design or component changes. Some failures may also result in safety issues requiring operational modifications until the offending elements under analysis can be corrected. If components or systems must be returned to maintenance facilities for corrective repairs, there will be operational and business impacts due to increased unavailability and unplanned transportation cost.

Applicable Methods & Tools

Operations manuals generally provide operators the steps and activities required to run the system.

Training and Certification

Adequate training must be provided for the operators who are required to operate the system. There are many objectives of training:

- Provide initial training for all operators in order to equip them with the skill and knowledge to operate the system. Ideally, this process will begin prior to system transition and will facilitate delivery of the system. It is important to define the certification standards and required training materials up front (for more information on material supply, please see Logistics).
- Provide continuation training to ensure currency of knowledge.

- Monitor the qualification/certification of the operators to ensure that all personnel operating the system meet the minimum skill requirements and that their currency remains valid.
- Monitor and evaluate the job performance to determine the adequacy of the training program.

Practical Considerations

The operation process sustains system services by assigning trained personnel to operate the system, as well as by monitoring operator-system performance and monitoring the system performance. In order to sustain services, the operation process identifies and analyzes operational problems in relation to agreements, stakeholder requirements, and organizational constraints. When the system replaces an existing system, it may be necessary to manage the migration between systems such that persistent stakeholders do not experience a breakdown in services.

Results of a successful implementation of the operation process include

- an operation strategy is defined and refined along the way
- services that meet stakeholder requirements are delivered
- approved, corrective action requests are satisfactorily completed
- stakeholder satisfaction is maintained

Outputs of the operation process include

- operational strategy, including staffing and sustainment of enabling systems and materials
- system performance reports (statistics, usage data, and operational cost data)
- system trouble/anomaly reports with recommendations for appropriate action
- operational availability constraints to influence future design and specification of similar systems or reused system elements

Activities of the operation process include

- provide operator training to sustain a pool of operators
- track system performance and account for operational availability
- perform operational analysis
- manage operational support logistics
- document system status and actions taken
- report malfunctions and recommendations for improvement

References

Works Cited

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th Edition. Englewood Cliffs, NJ, USA:Prentice Hall.

Institute of Engineers Singapore. 2009. *Systems Engineering Body of Knowledge*. Provisional version 2.0. Singapore: Institute of Engineers Singapore.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Maintenance

System Maintenance planning begins early in the acquisition process with development of a maintenance concept. Maintenance planning is conducted to evolve and establish requirements and tasks to be accomplished for achieving, restoring, and maintaining operational capability for the life of the system. For a system to be sustained throughout its system life cycle, the maintenance process has to be executed concurrently with the operations process (ISO/IEC/IEEE 15288 2015, Clause 6.4.9).

Overview

The initial requirements for maintenance have to be defined during the stakeholder needs and requirement definition process (Clause 6.4.1) (ISO/IEC/IEEE 15288 2015) and continue to evolve during the development and operation of the system. Considerations include:

- Maximizing system availability to meet the operational requirements. This has to take into account the designed-in reliability and maintainability of the system and resources available.
- Preserving system operating potential through proper planning of system scheduled maintenance. This requires a reliability-centered maintenance strategy that incorporates preventive maintenance in order to preempt failures, thereby extending the mean time between corrective maintenance, as well as enhancing the availability of the system.
- Segmentation of maintenance activities for potential outsourcing of non-critical activities to approved maintenance subcontractors as to optimize scarce technical manpower resources and maintenance/repair turn-around times.
- Harnessing IT technology for maintenance management. This involves rigorous and systematic capturing and tracking of operating and maintenance activities to facilitate analysis and planning.

Maintenance management is concerned with the development and review of maintenance plans, as well as securing and coordinating resources, such as budget, service parts provisioning, and management of supporting tasks (e.g., contract administration, engineering support, and quality assurance). Maintenance planning relies on level of repair

analysis (LORA) as a function of the system acquisition process. Initial planning addresses actions and support necessary to ensure a minimum life cycle cost (LCC).

Process Approaches

The purpose of the maintenance process is to sustain the capability of a system to provide a service. This process monitors the system's capability to deliver services, records problems for analysis, takes corrective, adaptive, perfective, and preventive actions, and confirms restored capability. As a result of the successful implementation of the maintenance process

- a maintenance strategy is developed
- maintenance constraints are provided as inputs to requirements
- replacement system elements are made available
- services meeting stakeholder requirements are sustained
- the need for corrective design changes is reported
- failure and lifetime data is recorded

The project should implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the maintenance process:

- scheduled servicing, such as daily inspection/checks, servicing, and cleaning
- unscheduled servicing (carrying out fault detection and isolation to the faulty replaceable unit and replacement of the failed unit)
- re-configuration of the system for different roles or functions
- scheduled servicing (higher level scheduled servicing but below depot level)
- unscheduled servicing (carrying out more complicated fault isolation to the faulty replaceable unit and replacement of the failed unit)
- minor modifications
- minor damage repairs
- major scheduled servicing (e.g., overhaul and corrosion treatment)
- major repairs (beyond normal removal and replacement tasks)

The maintenance plan specifies the scheduled servicing tasks and intervals (preventive maintenance) and the unscheduled servicing tasks (adaptive or corrective maintenance). Tasks in the maintenance plan are allocated to the various maintenance agencies. A maintenance allocation chart is developed to tag the maintenance tasks to the appropriate maintenance agencies. These include: in-service or in-house work centers, approved contractors, affiliated maintenance or repair facilities, original equipment manufacturer (OEMs), etc. The maintenance plan also establishes the requirements for the support resources.

Related activities such as resource planning, budgeting, performance monitoring, upgrades, longer term supportability, and sustenance also need to be managed. These activities are being planned, managed, and executed over a longer time horizon and they concern the well being of the system over the entire life cycle.

Proper maintenance of the system (including maintenance-free system designs) relies very much on the availability of support resources, such as support and test equipment (STE), technical data and documentation, personnel, spares, and facilities. These have to be factored in during the acquisition agreement process.

Training and Certification

Adequate training must be provided for the technical personnel maintaining the system. While initial training may have been provided during the deployment phase, additional personnel may need to be trained to cope with the increased number of systems being fielded, as well as to cater to staff turnover. Timely updates to training materials and trained personnel may be required as part of system upgrades and evolution. It is important to define the certification standards and contract for the training materials as part of the supply agreement.

Practical Considerations

The organization responsible for maintaining the system should have clear thresholds established to determine whether a change requested by end users, changes to correct latent defects, or changes required to fulfill the evolving mission are within the scope of a maintenance change or require a more formal project to step through the entire systems engineering life-cycle. Evaluation criteria to make such a decision could include cost, schedule, risk, or criticality characteristics.

References

Works Cited

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th Edition. Upper Saddle River, NJ, USA: Prentice Hall.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense. February 19, 2010.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Institute of Engineers Singapore. 2009. *Systems Engineering Body of Knowledge,* Provisional version 2.0. Singapore: Institute of Engineers Singapore.

IISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Logistics

There are several definitions for logistics within systems engineering (SE) and the definition used will determine what activities are considered part of logistics. The SEBoK defines logistics as the science of planning and implementing the acquisition and use of the resources necessary to sustain the operation of a system.

Overview

The ability to *sustain the operation of a system* is determined by the inherent supportability of the system (a function of design) and the processes used to sustain the functions and capabilities of the system in the context of the end user. Figure 1, below, shows a Defense Acquisition University (DAU) model of the SE aspects for consideration in logistics and logistics planning (DAU 2010).

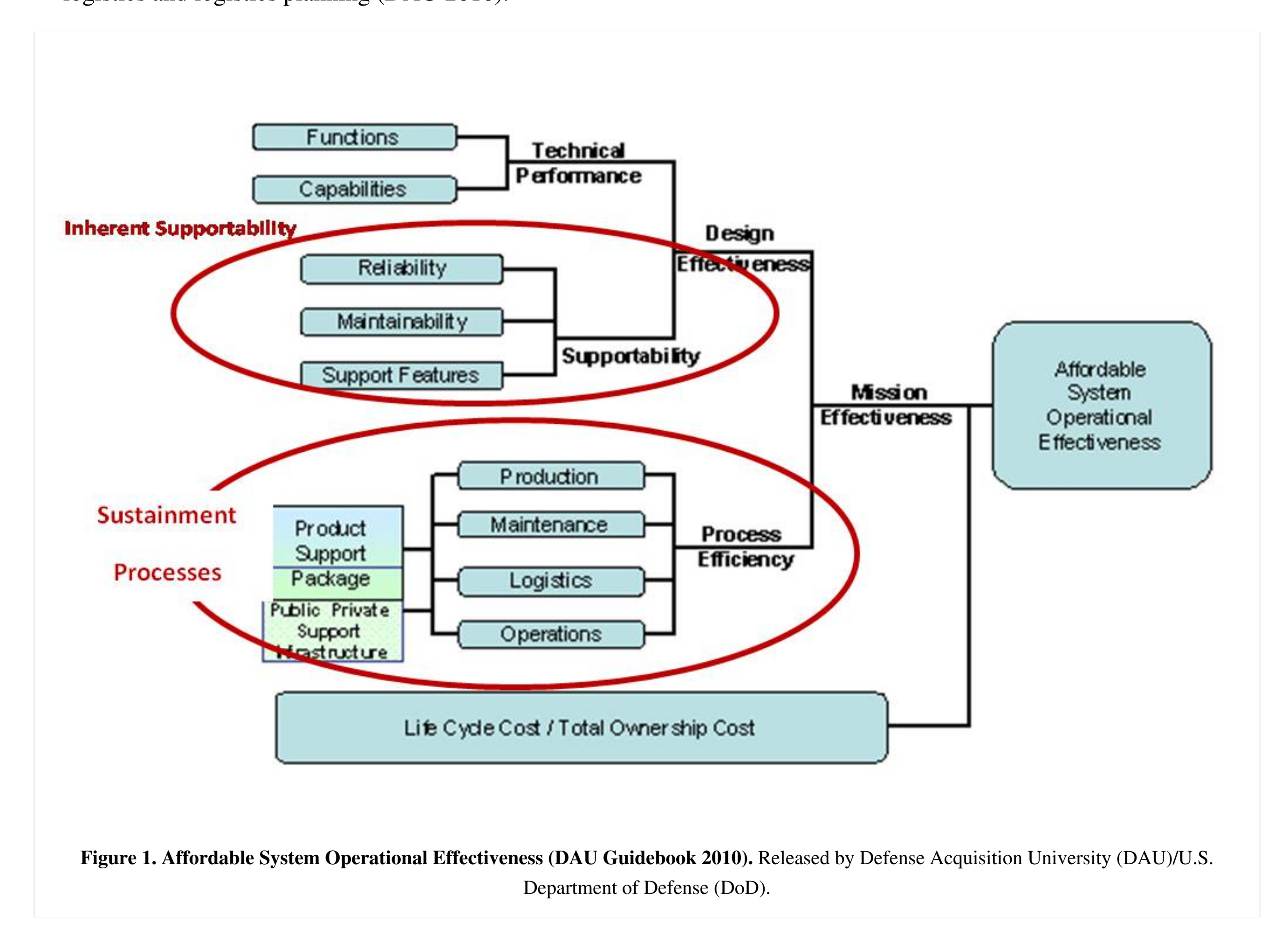


Figure 1. Affordable System Operational Effectiveness (DAU Guidebook 2010). Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Sustainment Planning

The focus of sustainment planning is to influence the inherent supportability of the system and to plan the sustainment capabilities and processes that will be used to sustain system operations.

Influence Inherent Supportability (Operational Suitability)

Sustainment influence requires an understanding of the concept of operations (ConOps), system missions, mission profiles, and system capabilities to understand the rationale behind functional and performance priorities. Understanding the rationale paves the way for decisions about necessary tradeoffs between system performance, availability, and life cycle cost (LCC), with impact on the cost effectiveness of system operation, maintenance, and

logistics support. There is no single list of sustainment considerations or specific way of grouping them as they are highly inter-related. They range from: compatibility, interoperability, transportability, reliability, maintainability, manpower, human factors, safety, natural environment effects (including occupational health, habitability, see Environmental Engineering); diagnostics & prognostics (including real-time maintenance data collection), and corrosion protection & mitigation. The following are key design considerations:

- **Architecture Considerations** - The focus on openness, modularity, scalability, and upgradeability is critical to implementing an incremental acquisition strategy. In addition, the architecture attributes that expand system flexibility and affordability can pay dividends later when obsolescence and end-of-life issues are resolved through a concerted technology refreshment strategy. Trade-offs are often required relative to the extent each attribute is used.
- **Reliability Considerations**: - Reliability is critical because it contributes to a system's effectiveness as well as its suitability in terms of logistics burden and the cost to fix failures. For each system, there is a level of basic reliability that must be achieved for the system to be considered useful. Reliability is also one of the most critical elements in determining the logistics infrastructure and footprint. Consequently, system reliability should be a primary focus during design (along with system technical performance, functions, and capabilities). The primary objective is to achieve the necessary probability of operational success and minimize the risk of failure within defined availability, cost, schedule, weight, power, and volume constraints. While performing such analyses, trade-offs should be conducted and dependencies should be explored with system maintainability and integrated with the supportability analysis that addresses support event frequency (i.e. reliability), event duration, and event cost. Such a focus will play a significant role in minimizing the necessary logistics footprint, while maximizing system availability.
- **Maintainability Considerations** - The design emphasis on maintainability is to reduce the maintenance burden and supply chain by reducing the time, personnel, tools, test equipment, training, facilities and cost to maintain the system. Maintainability engineering includes the activities, methods, and practices used to design minimal system maintenance requirements (designing out unnecessary and inefficient processes) and associated costs for preventive and corrective maintenance as well as servicing or calibration activities. Maintainability should be a designed-in capability and not an add-on option because good maintenance procedures cannot overcome poor system and equipment maintainability design. The primary objective is to reduce the time it takes for a properly trained maintainer to detect and isolate the failure (coverage and efficiency) and affect repair. Intrinsic factors contributing to maintainability are
    - **Modularity** - Packaging of components such that they can be repaired via remove and replace action vs. on-board repair. Care should be taken not to *over modularize* and trade-offs to evaluate replacement, transportation, and repair costs should be accomplished to determine the most cost effective approach.
    - **Interoperability** - The compatibility of components with standard interface protocols to facilitate rapid repair and enhancement/upgrade through black box technology using common interfaces. Physical interfaces should be designed so that mating between components can only happen correctly.
    - **Physical accessibility** - The designed-in structural assurance that components which require more frequent monitoring, checkout, and maintenance can be easily accessed. This is especially important in low observable platforms. Maintenance points should be directly visible and accessible to maintainers, including access for corrosion inspection and mitigation.
    - Designs that require *minimum preventative maintenance* including corrosion prevention and mitigation. Emphasis should be on balancing the maintenance requirement over the life cycle with minimal user workload.
    - **Embedded training and testing** when it is determined to be the optimal solution from a total ownership cost (TOC) and materiel availability perspective.
    - **Human Systems Integration (HSI)** to optimize total system performance and minimize life-cycle costs by designing systems and incorporating technologies that (a) require minimal manpower, (b) provide effective training, (c) can be operated and maintained by users, (d) are suitable (habitable and safe with minimal

environmental and occupational health hazards), and (e) are survivable (for both the user and the equipment).

- **Support Considerations** - Support features cannot be easily *added-on* after the design is established. Consequently, supportability should be a high priority early in the program's planning and integral to the system design and development process. Support features cut across reliability, maintainability, and the supply chain to facilitate detection, isolation, and timely repair/replacement of system anomalies. These include features for servicing and other activities necessary for operation and support including resources that contribute to the overall support of the system. Typical supportability features include diagnostics, prognostics (see CBM+ Guidebook), calibration requirements, many HSI issues (e.g. training, safety, HFE, occupational health, etc.), skill levels, documentation, maintenance data collection, compatibility, interoperability, transportability, handling (e.g., lift/hard/tie down points, etc.), packing requirements, facility requirements, accessibility, and other factors that contribute to an optimum environment for sustaining an operational system.

Planning Sustainment Processes

Process efficiency reflects how well the system can be produced, operated, serviced (including fueling) and maintained. It reflects the degree to which the logistics processes (including the supply chain), infrastructure, and footprint have been balanced to provide an agile, deployable, and operationally effective system.

Achieving process efficiency requires early and continuing emphasis on the various logistics support processes along with the design considerations. The continued emphasis is important because processes present opportunities for improving operational effectiveness even after the *design-in* window has passed via lean-six sigma, supply chain optimization, or other continuous process improvement (CPI) techniques.

Sustainment Analysis (Product Support Package)

The product support package documents the output of supportability analysis and includes details related to the following twelve elements (links below are to excerpts from (NATO RTO 2001):

- Product/information technology (IT) system/medical system support management (integrated life cycle sustainment planning)
  - product/IT system/medical system support strategies
  - life cycle sustainment planning
  - requirements management
  - total ownership costs (TOC)/life cycle costs (LCC) planning & management
  - Integration and management of product support activities
  - configuration management
  - production & distribution
  - energy, environmental, safety and health (EESH) management
  - policies & guidance
  - risk management
- Design Interface [1]
  - reliability
  - maintainability
  - supportability
  - affordability
  - configuration management
  - safety requirements
  - environmental and hazardous materials (HAZMAT) requirements
  - human systems integration (HSI)

    - calibration
    - anti-tamper
    - habitability
    - disposal
    - legal requirements
- Sustainment Engineering
    - failure reporting, analysis, and corrective action system (FRACAS)
    - value engineering
    - diminishing manufacturing sources and material shortages (DMSMS)
- Supply Support (materiel planning) [2]
- Maintenance Planning [3]
    - reliability centered maintenance (RCM)
    - maintenance concepts
    - levels of maintenance (level of repair analysis)
    - condition-based maintenance
    - prognostics & health management
- Support Equipment [4]
- Technical Data [5]
- Manpower & Personnel [6]
- Training & Training Support [7]
- Facilities & Infrastructure [8]
- Packaging, Handling, Storage, & Transportation [9]
- Computer Resources [10]

Sustainment Implementation

Once the system becomes operational, the results of sustainment planning efforts need to be implemented. SE supports the execution of the twelve integrated product support elements of a sustainment program that strives to ensure the system meets operational performance requirements in the most cost-effective manner over its total remaining life cycle, as illustrated in Figure 2.

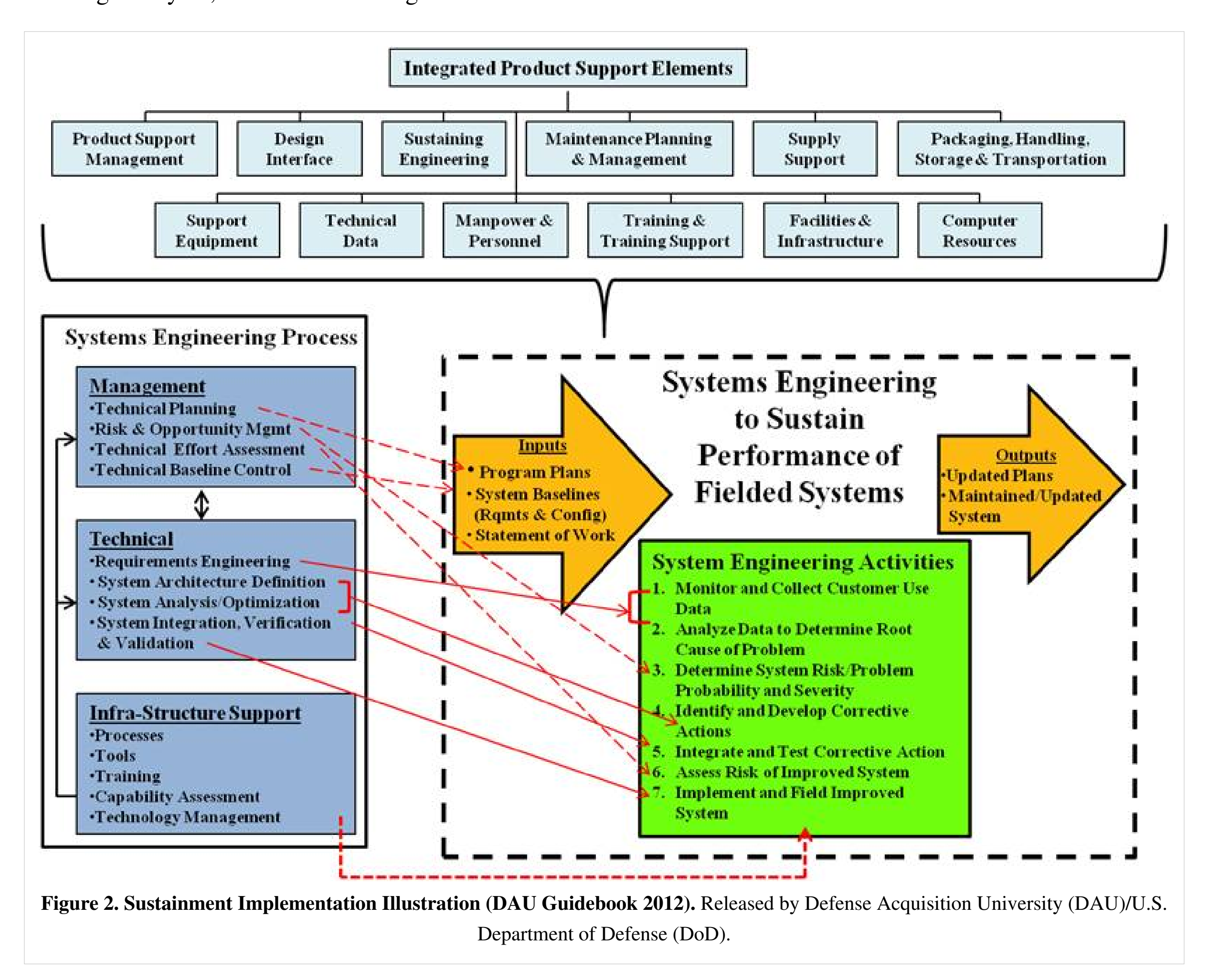


Figure 2. Sustainment Implementation Illustration (DAU Guidebook 2012). Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Once a system is put into use, SE is often required to correct problems that degrade continued use, and/or to add new capabilities to improve product performance in the current or a new environment. In the context of integrated product support, these SE activities correspond to the integrated product support (IPS) element *Sustaining Engineering*. Changes made to fielded systems to correct problems or increase performance should include any necessary adjustments to the IPS elements, and should consider the interrelationships and integration of the elements to maintain the effectiveness of system's support strategy.

The degree of change required to the product support elements varies with the severity of the problem. Minor problems may require a simple adjustment to a maintenance procedure, a change of supplier, a training course modification or a change to a technical manual. In contrast, problems that require system or component redesign may require engineering change proposals and approvals, IPS element trade studies, business case analysis, and updates to the product support strategy. The focus is to correct problems that degrade continued use, regardless of the degree of severity.

Evolutionary systems provide a strategy for acquisition of mature technology; the system delivers capabilities incrementally, planning for future capability enhancements. For these systems a system of systems (SoS) perspective is required to synchronize the primary and sustainment systems.

For more information refer to: *An Enterprise Framework for Operationally Effective System of Systems Design* (Bobinis and Herald 2012.).

References

Works Cited

Bobinis, J. and T. Herald. 2012. “An Enterprise Framework for Operationally Effective System of Systems Design.” *Journal of Enterprise Architecture.* 8(2), May 2012. Available at: https://www.mendling.com/publications/JEA12-2.pdf.

DAU. 2010. Defense Acquisition Guidebook (DAG). Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

NATO RTO. 2001. *Logistics Test and Evaluation in Flight Test*. Flight Test Techniques Series – Volume 20. Quebec, Canada: North Atlantic Treaty Organization (NATO) Research and Technology Organization (RTO). RTO-AG-300 Vol. 20, AC/323(SCI-010)TP/38. Table of contents available at: http://ftp.rta.nato.int/public//PubFullText/RTO/AG/RTO-AG-300-V20///AG-300-V20-$$TOC.pdf

Primary References

Blanchard, B.S. 1998. *Logistics Engineering and Management.* Upper Saddle River, NJ, USA: Prentice Hall.

Blanchard, B. and W. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th Ed. Englewood Cliffs, NJ, USA: Prentice-Hall.

Bobinis, J. and T. Herald. 2012. “An Enterprise Framework for Operationally Effective System of Systems Design.” *Journal of Enterprise Architecture.* 8(2), May 2012. Available at: https://www.mendling.com/publications/JEA12-2.pdf.

Daganzo, C. 2005. *Logistics Systems Analysis,* 4th Edition. New York, NY, USA: Springer.

Fabrycky, W.J. and B.S. Blanchard. 1991. *Life-Cycle Cost and Economic Analysis*. Upper Saddle River, NJ, USA: Prentice-Hall.

Ghiani, G., G. Laporte, and R. Musmanno. 2004. *Introduction to Logistics Systems Planning and Control.* Hoboken, NJ, USA: Wiley-Interscience.

Jones, J.V. 1995. *Integrated Logistics Support Handbook.* New York, NY, USA: McGraw Hill.

Additional References

Barros, L.L. 1998. "The Optimization of Repair Decision Using Life-Cycle Cost Parameters." *IMA Journal of Management Mathematics.* 9(4): 403.

Berkowitz, D., J.N. Gupta, J.T. Simpson, and J.B. McWilliams. 2005. *Defining and Implementing Performance-Based Logistics in Government*. Washington, DC, USA: Defense Technical Information Center. Accessed 6 Sept 2011. Available at: http://handle.dtic.mil/100.2/ADP018510.

Gajpal, P.P., L.S. Ganesh, and C. Rajendran. 1994. "Criticality Analysis of Spare Parts Using the Analytic Hierarchy Process." *International Journal of Production Economics.* 35(1-3): 293-297.

MITRE. 2011. "Integrated Logistics Support." *Systems Engineering Guide.* Accessed 11 March 2012 at [[11]].

Murthy, D.N.P. and W.R. Blischke. 2000. "Strategic Warranty Management: A Life-Cycle Approach." *Engineering Management.* 47(1): 40-54.

Northrop Grumman Corporation. 2000. *Logistics Systems Engineering*. Accessed 6 Sept 2011. Available at: http://www.northropgrumman.com/Capabilities/NavigationSystemsLogisticsSystemsEngineering/Documents/nsd_logistics.pdf.

Solomon, R., P.A. Sandborn, and M.G. Pecht. 2000. "Electronic Part Life Cycle Concepts and Obsolescence Forecasting." *IEEE Transactions on Components and Packaging Technologies.* 23(4): 707-717.

Spengler, T. and M. Schroter. 2003. "Strategic Management of Spare Parts in Closed-Loop Supply Chains: A System Dynamics Approach." *Interfaces.* p. 7-17.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-12.pdf
[2] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-06.pdf
[3] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-03.pdf
[4] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-05.pdf
[5] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-07.pdf
[6] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-04.pdf
[7] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-08.pdf
[8] http://www.decisionlens.com/docs/WP_Strategic_Facilities_and_Infrastructure_Planning.pdf
[9] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-11.pdf
[10] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-09.pdf
[11] http://www.mitre.org/work/systems_engineering/guide/acquisition_systems_engineering/integrated_logistics_support/

Knowledge Management: Systems Engineering Management

Systems Engineering Management

This knowledge area is about managing the resources and assets allocated to perform systems engineering, often in the context of a project or a service, but sometimes in the context of a less well-defined activity. Systems engineering management is distinguished from general project management by its focus on the technical or engineering aspects of a project. SEM also encompasses exploratory research and development (R&D) activities at the enterprise level in commercial or government operations.

Topics

Each part of the SEBoK is composed of knowledge areas (KAs). Each KA groups topics together around a theme related to the overall subject of the part. This KA contains the following topics:

- Planning
- Assessment and Control
- Risk Management
- Measurement
- Decision Management
- Configuration Management
- Information Management
- Quality Management

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

Discussion

Implementing systems engineering (SE) requires the coordination of technical and managerial endeavors. Success with the technical is not possible in the absence of the managerial. Management provides the planning, organizational structure, collaborative environment, and program controls to ensure that stakeholder needs are met.

The Venn diagram below provides some context for thinking about SEM. It shows that some functions are managed within the SE function, while others are managed in collaboration with the management of systems implementation and with overall project and systems management.

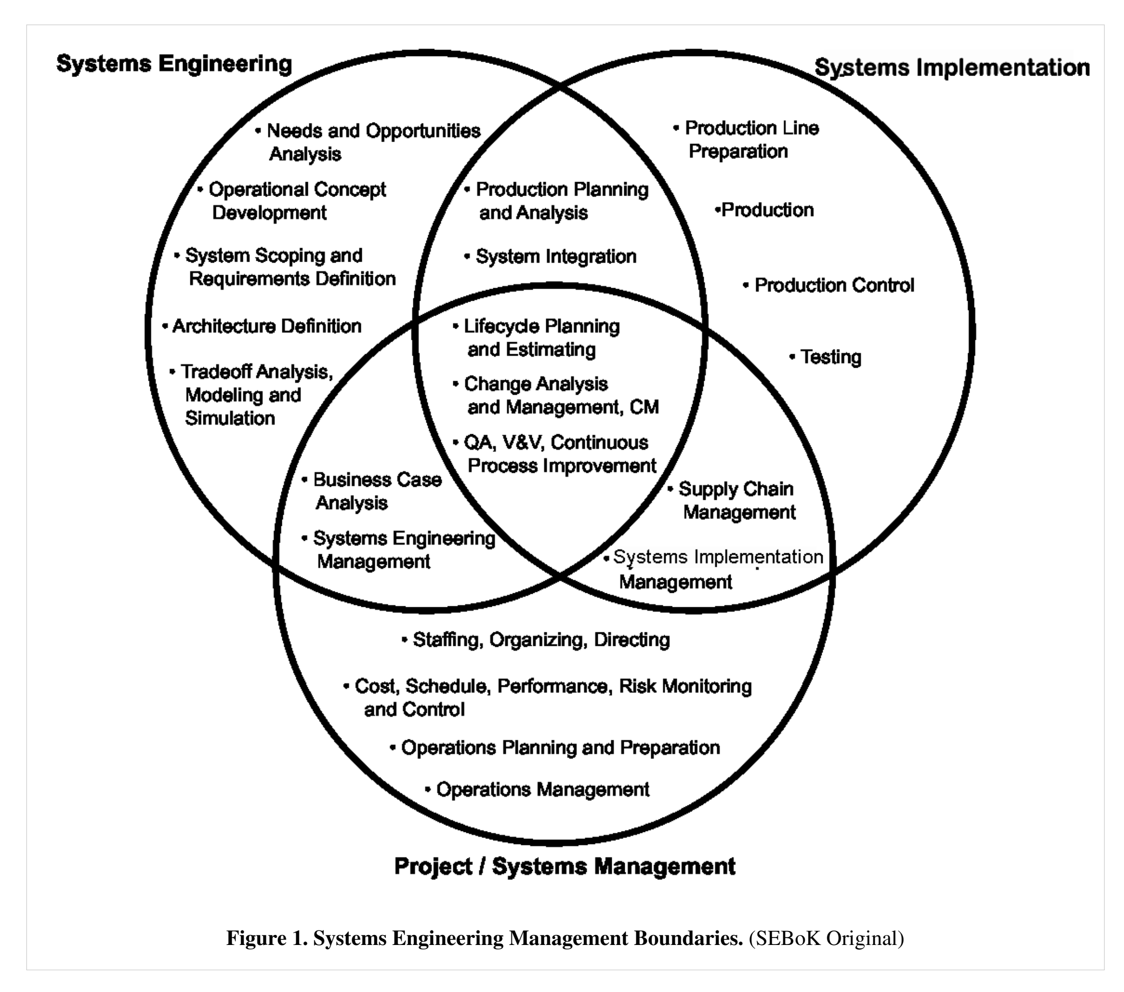


Figure 1. Systems Engineering Management Boundaries. (SEBoK Original)

There is no one-size-fits-all way to define the details of where SEM functions are performed. An in-company SE organization does not run its own accounting system, but relies on the corporate management organization for this aspect of SEM. A company performing only SE *does* include the accounting functions as part of SEM. In all cases, the managers of the SE function must be actively involved in the management of all the activities within the SE system boundary, including working out what collaborative arrangements best fit their situation. They must also remain aware of management events in their environment outside the system boundary that may affect their ability to perform. Part 6 of the SEBoK includes relevant knowledge areas for collaborative management, including Systems Engineering and Software Engineering, Systems Engineering and Project Management, Systems Engineering and Industrial Engineering, Systems Engineering and Procurement/Acquisition, and Systems Engineering and Specialty Engineering.

References

Works Cited

None.

Primary References

Blanchard, B.S. 2004. *Systems Engineering Management,* 3rd ed. New York, NY, USA: John Wiley & Sons Inc.

Sage, A.P. and W. Rouse. 2009. *Handbook of Systems Engineering and Management,* 2nd Ed. Hoboken, NJ, USA: John Wiley and Sons.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Planning

Planning is an important aspect of systems engineering management (SEM). Systems engineering (SE) planning is performed concurrently and collaboratively with project planning. It involves developing and integrating technical plans to achieve the technical project objectives within the resource constraints and risk thresholds. The planning involves the success-critical stakeholders to ensure that necessary tasks are defined with the right timing in the life cycle in order to manage acceptable risks levels, meet schedules, and avoid costly omissions.

SE Planning Process Overview

SE planning provides the following elements:

- Definition of the project from a technical perspective.
- Definition or tailoring of engineering processes, practices, methods, and supporting enabling environments to be used to develop products or services, as well as plans for transition and implementation of the products or services, as required by agreements.
- Definition of the technical organizational, personnel, and team functions and responsibilities, as well as all disciplines required during the project life cycle.
- Definition of the appropriate life cycle model or approach for the products or services.
- Definition and timing of technical reviews, product or service assessments, and control mechanisms across the life cycle, including the success criteria such as cost, schedule, and technical performance at identified project milestones.
- Estimation of technical cost and schedule based on the effort needed to meet the requirements; this estimation becomes input to project cost and schedule planning.
- Determination of critical technologies, as well as the associated risks and actions needed to manage and transition these technologies.
- Identification of linkages to other project management efforts.
- Documentation of and commitment to the technical planning.

Scope

SE planning begins with analyzing the scope of technical work to be performed and gaining an understanding the constraints, risks, and objectives that define and bound the solution space for the product or service. The planning includes estimating the size of the work products, establishing a schedule (or integrating the technical tasks into the project schedule), identification of risks, and negotiating commitments. Iteration of these planning tasks may be necessary to establish a balanced plan with respect to cost, schedule, technical performance, and quality. The planning continues to evolve with each successive life cycle phase of the project (NASA 2007, 1-360; SEI 1995, 12).

SE planning addresses all programmatic and technical elements of the project to ensure a comprehensive and integrated plan for all of the project's technical aspects and should account for the full scope of technical activities, including system development and definition, risk management, quality management, configuration management, measurement, information management, production, verification and testing, integration, validation, and deployment. SE planning integrates all SE functions to ensure that plans, requirements, operational concepts, and architectures are consistent and feasible.

The scope of planning can vary from planning a specific task to developing a major technical plan. The integrated planning effort will determine what level of planning and accompanying documentation is appropriate for the project.

Integration

The integration of each plan with other higher-level, peer, or subordinate plans is an essential part of SE planning. For the technical effort, the systems engineering management plan (SEMP), also frequently reffered to as the systems engineering plan (SEP), is the highest level technical plan. It is subordinate to the project plan and often has a number of subordinate technical plans providing detail on specific technical focus areas (INCOSE 2011, sec. 5.1.2.2; NASA 2007, appendix J).

In U.S. defense work, the terms SEP and SEMP are not interchangeable. The SEP is a high-level plan that is made before the system acquisition and development begins. It is written by the government customer. The SEMP is the specific development plan written by the developer (or contractor). In this context, intent, and content of these documents are quite different. For example, a SEP will have an acquisition plan that would not be included in a SEMP. Figure 1 below shows the SEMP and integrated plans.

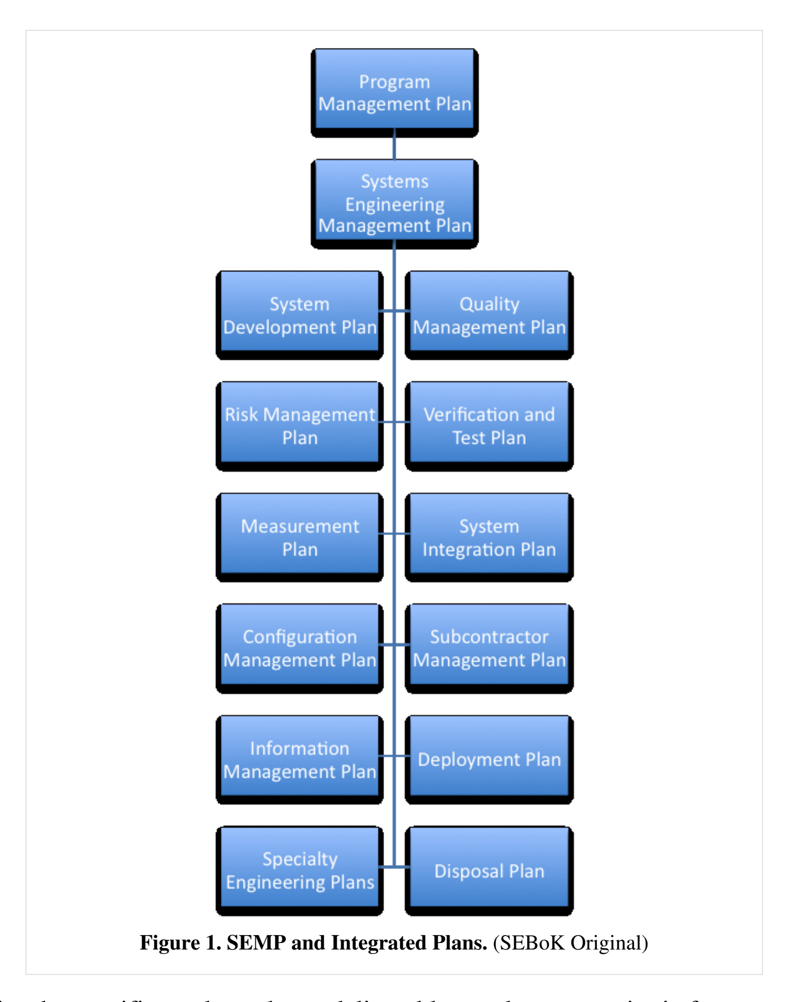


Figure 1. SEMP and Integrated Plans. (SEBoK Original)

Task planning identifies the specific work products, deliverables, and success criteria for systems engineering efforts in support of integrated planning and project objectives. The success criteria are defined in terms of cost, schedule, and technical performance at identified project milestones. Detailed task planning identifies specific resource requirements (e.g., skills, equipment, facilities, and funding) as a function of time and project milestones.

SE planning is accomplished by both the acquirer and supplier and the activities for SE planning are performed in the context of the respective enterprise. The activities establish and identify relevant policies and procedures for managing and executing the project management and technical effort, identifying the management and technical tasks, their interdependencies, risks, and opportunities, and providing estimates of needed resources/budgets. Plans are updated and refined throughout the development process based on status updates and evolving project requirements (SEI 2007).

Linkages to Other Systems Engineering Management Topics

The project planning process is closely coupled with the measurement, assessment and control, decision management, and risk management processes.

The measurement process provides inputs for estimation models. Estimates and other products from planning are used in decision management. SE assessment and control processes use planning results for setting milestones and assessing progress. Risk management uses the planning cost models, schedule estimates, and uncertainty distributions to support quantitative risk analysis (as desired).

Additionally, planning needs to use the outputs from assessment and control as well as risk management to ensure corrective actions have been accounted for in planning future activities. The planning may need to be updated based on results from technical reviews (from assessment and control) addressing issues pertaining to: measurement, problems that were identified during the performance of risk management activities, or decisions made as a result of

the decision management activities (INCOSE 2010, sec. 6.1).

Practical Considerations

Pitfalls

Some of the key pitfalls encountered in planning and performing SE planning are listed in Table 1.

Table 1. Major Pitfalls with Planning. (SEBoK Original)

Name	Description
Incomplete and Rushed Planning	Inadequate SE planning causes significant adverse impacts on all other engineering activities. Although one may be tempted to save time by rushing the planning, inadequate planning can create additional costs and interfere with the schedule due to planning omissions, lack of detail, lack of integration of efforts, infeasible cost and schedules, etc.
Inexperienced Staff	Lack of highly experienced engineering staff members, especially in similar projects, will likely result in inadequate planning. Less experienced engineers are often assigned significant roles in the SE planning; however, they may not have the appropriate judgment to lay out realistic and achievable plans. It is essential to assign the SE planning tasks to those with a good amount of relevant experience.

Good Practices

Some good practices gathered from the references are in Table 2.

Table 2. Proven Practices with Planning. (SEBoK Original)

Name	Description
Use Multiple Disciplines	Get technical resources from all disciplines involved in the planning process.
Early Conflict Resolution	Resolve schedule and resource conflicts early.
Task Independence	Tasks should be as independent as possible.
Define Interdependencies	Define task interdependencies, using dependency networks or other approaches.
Risk Management	Integrate risk management with the SE planning to identify areas that require special attention and/or trades.
Management Reserve	The amount of management reserve should be based on the risk associated with the plan.
Use Historical Data	Use historical data for estimates and adjust for differences in the project.
Consider Lead Times	Identify lead times and ensure that you account for them in the planning (e.g., the development of analytical tools).
Update Plans	Prepare to update plans as additional information becomes available or changes are needed.
Use IPDTs	An integrated product development team (IPDT) (or integrated product team (IPT)) is often useful to ensure adequate communication across the necessary disciplines, timely integration of all design considerations, as well as integration, testing, and consideration of the full range of risks that need to be addressed. Although there are some issues that need to be managed with them, IPDTs tend to break down the communication and knowledge stovepipes that often exist.

Additional good practices can be found in the *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)*, *NASA Systems Engineering Handbook*, the *INCOSE Systems Engineering Handbook*, and *Systems and Software Engineering - Life Cycle Processes - Project Management* (Caltrans and USDOT 2005, 278; NASA December 2007, 1-360, sec. 6.1; INCOSE 2011, sec. 5.1; ISO/IEC/IEEE 2009, Clause 6.1).

References

Works Cited

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense, February 19.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

NASA. 2007. *NASA Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

SEI. 1995. *A systems engineering capability maturity model.* Version 1.1. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-95-MM-003.

Primary References

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

NASA. 2007. *NASA Systems Engineering Handbook.* Washington, D.C., USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

SEI. 1995. *A Systems Engineering Capability Maturity Model,* version 1.1. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-95-MM-003.

SEI. 2007. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.2, measurement and analysis process area. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Additional References

Boehm, B., C. Abts, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D.J. Reifer, B. Steece. 2000. *Software Cost Estimation with COCOMO II*. Englewood Cliffs, NJ, USA: Prentice Hall

DeMarco, T. and T. Lister. 2003. *Waltzing with Bears; Managing Risks on Software Projects.* New York, NY, USA: Dorset House.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

Valerdi, R. 2008. *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort in Complex Systems*. Saarbrücken,Germany: VDM Verlag Dr. Muller

---

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Assessment and Control

The purpose of systems engineering assessment and control (SEAC) is to provide adequate visibility into the project's actual technical progress and risks with respect to the technical plans (i.e., systems engineering management plan (SEMP) or systems engineering plan (SEP) and subordinate plans). The visibility allows the project team to take timely preventive action when disruptive trends are recognized or corrective action when performance deviates beyond established thresholds or expected values. SEAC includes preparing for and conducting reviews and audits to monitor performance. The results of the reviews and measurement analyses are used to identify and record findings/discrepancies and may lead to causal analysis and corrective/preventive action plans. Action plans are implemented, tracked, and monitored to closure. (NASA 2007, Section 6.7; SEG-ITS, 2009, Section 3.9.3, 3.9.10; INCOSE, 2010, Clause 6.2; SEI, 2007)

Systems Engineering Assessment and Control Process Overview

The SEAC process involves determining and initiating the appropriate handling strategies and actions for findings and/or discrepancies that are uncovered in the enterprise, infrastructure, or life cycle activities associated with the project. Analysis of the causes of the findings/discrepancies aids in the determination of appropriate handling strategies. Implementation of approved preventive, corrective, or improvement actions ensures satisfactory completion of the project within planned technical, schedule, and cost objectives. Potential action plans for findings and/or discrepancies are reviewed in the context of the overall set of actions and priorities in order to optimize the benefits to the project and/or organization. Interrelated items are analyzed together to obtain a consistent and cost-effective resolution.

The SEAC process includes the following steps:

- monitor and review technical performance and resource use against plans
- monitor technical risk, escalate significant risks to the project risk register and seek project funding to execute risk mitigation plans
- hold technical reviews and report outcomes at the project reviews
- analyze issues and determine appropriate actions
- manage actions to closure
- hold a post-delivery assessment (also known as a post-project review) to capture knowledge associated with the project (this may be a separate technical assessment or it may be conducted as part of the project assessment and control process).

The following activities are normally conducted as part of a project assessment and control process:

- authorization, release and closure of work
- monitor project performance and resource usage against plan
- monitor project risk and authorize expenditure of project funds to execute risk mitigation plans
- hold project reviews
- analyze issues and determine appropriate actions
- manage actions to closure
- hold a post-delivery assessment (also known as a post-project review) to capture knowledge associated with the project

Examples of major technical reviews used in SEAC are shown in Table 1 from DAU (2010).

Table 1. Major Technical Review Examples (DAU 2012). Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Name	Description
Alternative Systems Review	A multi-disciplined review to ensure the resulting set of requirements agrees with the customers' needs and expectations.
Critical Design Review (CDR)	A multi-disciplined review establishing the initial product baseline to ensure that the system under review has a reasonable expectation of satisfying the requirements of the capability development document within the currently allocated budget and schedule.
Functional Configuration Audit	Formal examination of the as tested characteristics of a configuration item (hardware and software) with the objective of verifying that actual performance complies with design and interface requirements in the functional baseline.
In-Service Review	A multi-disciplined product and process assessment that is performed to ensure that the system under review is operationally employed with well-understood and managed risk.
Initial Technical Review	A multi-disciplined review that supports a program's initial program objective memorandum submission.
Integrated Baseline Review	A joint assessment conducted by the government program manager and the contractor to establish the performance measurement baseline.
Operational Test Readiness Review	A multi-disciplined product and process assessment to ensure that the system can proceed into initial operational test and evaluation with a high probability of success, and also that the system is effective and suitable for service introduction.
Production Readiness Review (PRR)	The examination of a program to determine if the design is ready for production and if the prime contractor and major subcontractors have accomplished adequate production planning without incurring unacceptable risks that will breach thresholds of schedule, performance, cost, or other established criteria.
Physical Configuration Audit	An examination of the actual configuration of an item being produced around the time of the full-rate production decision.
Preliminary Design Review (PDR)	A technical assessment establishing the physically allocated baseline to ensure that the system under review has a reasonable expectation of being judged operationally effective and suitable.
System Functional Review (SFR)	A multi-disciplined review to ensure that the system's functional baseline is established and has a reasonable expectation of satisfying the requirements of the initial capabilities document or draft capability development document within the currently allocated budget and schedule.
System Requirements Review (SRR)	A multi-disciplined review to ensure that the system under review can proceed into initial systems development and that all system requirements and performance requirements derived from the initial capabilities document or draft capability development document are defined and testable, as well as being consistent with cost, schedule, risk, technology readiness, and other system constraints.
System Verification Review (SVR)	A multi-disciplined product and process assessment to ensure the system under review can proceed into low-rate initial production and full-rate production within cost (program budget), schedule (program schedule), risk, and other system constraints.

Technology Readiness Assessment	A systematic, metrics-based process that assesses the maturity of critical technology elements, such as sustainment drivers.
Test Readiness Review (TRR)	A multi-disciplined review designed to ensure that the subsystem or system under review is ready to proceed into formal testing.

Linkages to Other Systems Engineering Management Topics

The SE assessment and control process is closely coupled with the measurement, planning, decision management, and risk management processes. The measurement process provides indicators for comparing actuals to plans. planning provides estimates and milestones that constitute plans for monitoring as well as the project plan, which uses measurements to monitor progress. Decision management uses the results of project monitoring as decision criteria for making control decisions.

Practical Considerations

Key pitfalls and good practices related to SEAC are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing SE assessment and control are shown in Table 2.

Table 2. Major Pitfalls with Assessment and Control. (SEBoK Original)

Name	Description
No Measurement	Since the assessment and control activities are highly dependent on insightful measurement information, it is usually ineffective to proceed independently from the measurement efforts - what you get is what you measure.
"Something in Time" Culture	Some things are easier to measure than others - for instance, delivery to cost and schedule. Don't focus on these and neglect harder things to measure like quality of the system, Avoid a "something in time" culture where meeting the schedule takes priority over everything else, but what is delivered is not fit for purpose, which results in the need to rework the project.
No Teeth	Make sure that the technical review gates have "teeth". Sometimes the project manager is given authority (or can appeal to someone with authority) to over-ride a gate decision and allow work to proceed, even when the gate has exposed significant issues with the technical quality of the system or associated work products. This is a major risk if the organization is strongly schedule-driven; it can't afford the time to do it right, but somehow it finds the time to do it again (rework).
Too Early Baselining	Don't baseline requirements or designs too early. Often there is strong pressure to baseline system requirements and designs before they are fully understood or agreed, in order to start subsystem or component development. This just guarantees high levels of rework.

Good Practices

Some good practices gathered from the references are shown in Table 3.

Table 3. Proven Practices with Assessment and Control. (SEBoK Original)

Name	Description
Independence	Provide independent (from customer) assessment and recommendations on resources, schedule, technical status, and risk based on experience and trend analysis.
Peer Reviews	Use peer reviews to ensure the quality of a products work before they are submitted for gate review.
Accept Uncertainty	Communicate uncertainties in requirements or designs and accept that uncertainty is a normal part of developing a system.
Risk Mitigation Plans	Do not penalize a project at gate review if they admit uncertainty in requirements - ask for their risk mitigation plan to manage the uncertainty.
Just In-Time Baselining	Baseline requirements and designs only when you need to - when other work is committed based on the stability of the requirement or design. If work has to start and the requirement or design is still uncertain, consider how you can build robustness into the system to handle the uncertainty with minimum rework.
Communication	Document and communicate status findings and recommendations to stakeholders.
Full Visibility	Ensure that action items and action-item status, as well as other key status items, are visible to all project participants.
Leverage Previous Root Cause Analysis	When performing root cause analysis, take into account the root cause and resolution data documented in previous related findings/discrepancies.
Concurrent Management	Plan and perform assessment and control concurrently with the activities for Measurement and Risk Management.
Lessons Learned and Post-Mortems	Hold post-delivery assessments or post-project reviews to capture knowledge associated with the project – e.g., to augment and improve estimation models, lessons learned databases, gate review checklists, etc.

Additional good practices can be found in INCOSE (2010, Clause 6.2), SEG-ITS (2009, Sections 3.9.3 and 3.9.10), INCOSE (2010, Section 5.2.1.5), and NASA (2007, Section 6.7).

References

Works Cited

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

SEI. 2007. "Measurement and Analysis Process Area," in *Capability Maturity Model Integrated (CMMI) for Development,* version 1.2. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Primary References

Caltrans and USDOT. 2005. *[[Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)],]* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105

SEI. 2007. "Measurement and Analysis Process Area," in *Capability Maturity Model Integrated (CMMI) for Development,* version 1.2. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Additional References

ISO/IEC/IEEE. 2009. *ISO/IEC/IEEE 16326|Systems and Software Engineering - Life Cycle Processes - Project Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Risk Management

The purpose of risk management is to reduce potential risks to an acceptable level before they occur, throughout the life of the product or project. Risk management is a continuous, forward-looking process that is applied to anticipate and avert risks that may adversely impact the project, and can be considered both a project management and a systems engineering process. A balance must be achieved on each project in terms of overall risk management ownership, implementation, and day-to-day responsibility between these two top-level processes.

For the SEBoK, risk management falls under the umbrella of Systems Engineering Management, though the wider body of risk literature is explored below.

Risk Management Process Overview

Risk is a measure of the potential inability to achieve overall program objectives within defined cost, schedule, and technical constraints. It has the following two components (DAU 2003a):

1. the probability (or likelihood) of failing to achieve a particular outcome
2. the consequences (or impact) of failing to achieve that outcome

In the domain of catastrophic risk analysis, risk has three components: (1) threat, (2) vulnerability, and (3) consequence (Willis et al. 2005).

Risk management involves defining a risk management strategy, identifying and analyzing risks, handling selected risks, and monitoring the progress in reducing risks to an acceptable level (SEI 2010; DoD 2015; DAU 2003a; DAU 2003b; PMI 2013) (Opportunity and opportunity management is briefly discussed in below).

The SE risk management process includes the following activities:

- risk planning
- risk identification
- risk analysis
- risk handling
- risk monitoring

ISO/IEC/IEEE 16085 provides a detailed set of risk management activities and tasks which can be utilized in a risk management process aligned with ISO 31000:2009, Risk management — Principles and Guidelines, and ISO Guide 73:2009,

Risk management — Vocabulary. ISO 9001:2008 standard provides risk-based preventive action requirements in subclause 8.5.3.

The Risk Management Process section of the INCOSE Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities, 4th Edition, provides a comprehensive overview of risk management which is intended to be consistent with the Risk Management Process section of ISO 15288.

Risk Planning

Risk planning establishes and maintains a strategy for identifying, analyzing, handling, and monitoring risks within the project. The strategy, both the process and its implementation, is documented in a risk management plan (RMP).

The risk management process and its implementation should be tailored to each project, updated as appropriate throughout the life of the project and the RMP should be transmitted in an appropriate means to the project team and key stakeholders.

As necessary the risk management strategy includes the risk management process of all supply chain suppliers and describes how risks from all suppliers will be raised to the next level(s) for incorporation in the project risk process.

The context of the Risk Management process should include a description of stakeholders' perspectives, risk categories, and a description (perhaps by reference) of the technical and managerial objectives, assumptions and constraints. The risk categories include the relevant technical areas of the system and facilitate identification of risks across the life cycle of the system. As noted in ISO 31000 the aim of this step is to generate a comprehensive list of risks based on those events that might create, enhance, prevent, degrade, accelerate or delay the achievement of objectives.

The RMP should contain key risk management information; Conrow (2003) identifies the following as key components of RMP:

- a project summary
- project acquisition and contracting strategies
- key definitions
- a list of key documents
- process steps
- inputs, tools and techniques, and outputs per process step
- linkages between risk management and other project processes
- key ground rules and assumptions
- risk categories
- seller and buyer roles and responsibilities
- organizational and personnel roles and responsibilities

Generally, the level of detail in a RMP is risk-driven, with simple plans for low risk projects and detailed plans for high risk projects.

Risk Identification

Risk identification is the process of examining the project products, processes, and requirements to identify and document candidate risks. Risk identification should be performed continuously at the individual-level as well as through formerly structured events at both regular intervals and following major program changes (e.g., project initiation, re-baselining, change in acquisition phase, etc.).

Conrow (2009) states that systems engineers should use one or more top-level approaches (e.g., work breakdown structure (WBS), key processes evaluation, key requirements evaluation, etc.) and one or more lower-level approaches (e.g., affinity, brainstorming, checklists and taxonomies, examining critical path activities, expert judgment, Ishikawa diagrams, etc.) in risk identification. For example, lower-level checklists and taxonomies exist for software risk identification (Conrow and Shishido 1997, 83-89, p. 84; Boehm 1989, 115-125, Carr et al. 1993, p. A-2) and operational risk identification (Gallagher et al. 2005, p. 4), and have been used on a wide variety of programs. The top and lower-level approaches are essential but there is no single accepted method — all approaches should be examined and used as appropriate.

Candidate risk documentation should include the following items where possible, as identified by Conrow (2003 p.198):

- risk title
- structured risk description
- applicable risk categories
- potential root causes
- relevant historical information
- responsible individual and manager

It is important to use structured risk descriptions such as an *if-then* format: *if* (an event occurs--trigger), *then* (an outcome or affect occurs). Another useful construct is a *condition* (that exists) that leads to a potential *consequence* (outcome) (Gluch 1994). These approaches help the analyst to better think through the potential nature of the risk.

Risk analysis and risk handling activities should only be performed on approved risks to ensure the best use of scarce resources and maintain focus on the correct risks.

Risk Analysis

Risk analysis is the process of systematically evaluating each identified, approved risk to estimate the probability of occurrence (likelihood) and consequence of occurrence (impact), and then converting the results to a corresponding risk level or rating.

There is no *best* analysis approach for a given risk category. Risk scales and a corresponding matrix, simulations, and probabilistic risk assessments are often used for technical risks, while decision trees, simulations and payoff matrices are used for cost risk; and simulations are used for schedule risk. Risk analysis approaches are sometimes grouped into qualitative and quantitative methods. A structured, repeatable methodology should be used in order to increase analysis accuracy and reduce uncertainty over time.

The most common qualitative method (typically) uses ordinal probability and consequence scales coupled with a risk matrix (also known as a risk cube or mapping matrix) to convert the resulting values to a risk level. Here, one or more probability of occurrence scales, coupled with three consequences of occurrence scales (cost, performance, schedule) are typically used. Mathematical operations should not be performed on ordinal scale values to prevent erroneous results (Conrow 2003, p. 187-364).

Once the risk level for each risk is determined, the risks need to be prioritized. Prioritization is typically performed by risk level (e.g., low, medium, high), risk score (the pair of max (probability), max (consequence) values), and other considerations such as time-frame, frequency of occurrence, and interrelationship with other risks (Conrow 2003, pp. 187-364). An additional prioritization technique is to convert results into an estimated cost, performance, and schedule value (e.g., probability budget consequence). However, the result is only a point estimate and not a distribution of risk.

Widely used quantitative methods include decision trees and the associated expected monetary value analysis (Clemen and Reilly 2001), modeling and simulation (Law 2007; Mun 2010; Vose 2000), payoff matrices (Kerzner 2009, p. 747-751), probabilistic risk assessments (Kumamoto and Henley 1996; NASA 2002), and other techniques. Risk prioritization can directly result from the quantitative methods employed. For quantitative approaches, care is needed in developing the model structure, since the results will only be as good as the accuracy of the structure, coupled with the characteristics of probability estimates or distributions used to model the risks (Law 2007; Evans, Hastings, and Peacock 2011).

If multiple risk facets exist for a given item (e.g., cost risk, schedule risk, and technical risk) the different results should be integrated into a cohesive three-dimensional *picture* of risk. Sensitivity analyses can be applied to both qualitative and quantitative approaches in an attempt to understand how potential variability will affect results. Particular emphasis should be paid to compound risks (e.g., highly coupled technical risks with inadequate fixed budgets and schedules).

Risk Handling

Risk handling is the process that identifies and selects options and implements the desired option to reduce a risk to an acceptable level, given program constraints (budget, other resources) and objectives (DAU 2003a, 20-23, 70-78).

For a given system-of-interest (SoI), risk handling is primarily performed at two levels. At the system level, the overall ensemble of system risks is initially determined and prioritized and second-level draft risk element plans (REP's) are prepared for handling the risks. For more complex systems, it is important that the REP's at the higher SoI level are kept consistent with the system RMPs at the lower SoI level, and that the top-level RMP preserves continuing risk traceability across the SoI.

The risk handling strategy selected is the combination of the most desirable risk handling option coupled with a suitable implementation approach for that option (Conrow 2003). Risk handling options include assumption, avoidance, control (mitigation), and transfer. All four options should be evaluated and the best one chosen for each risk. An appropriate implementation approach is then chosen for that option. Hybrid strategies can be developed that include more than one risk handling option, but with a single implementation approach. Additional risk handling strategies can also be developed for a given risk and either implemented in parallel with the primary strategy or be made a contingent that is implemented if a particular trigger event occurs during the execution of the primary strategy. Often, this choice is difficult because of uncertainties in the risk probabilities and impacts. In such cases, buying information to reduce risk uncertainty via prototypes, benchmarking, surveying, modeling, etc. will clarify risk handling decisions (Boehm 1981).

Risk Handling Plans

A risk handling plan (RHP - a REP at the system level), should be developed and implemented for all *high* and *medium* risks and selected *low* risks as warranted.

As identified by Conrow (2003, 365-387), each RHP should include:

- a risk owner and management contacts
- selected option
- implementation approach
- estimated probability and consequence of occurrence levels at the start and conclusion of each activity
- specific measurable exit criteria for each activity
- appropriate metrics
- resources needed to implement the RHP

Metrics included in each RHP should provide an objective means of determining whether the risk handling strategy is on track and whether it needs to be updated. On larger projects these can include earned value, variation in schedule and technical performance measures (TPMs), and changes in risk level vs. time.

The activities present in each RHP should be integrated into the project’s integrated master schedule or equivalent; otherwise there will be ineffective risk monitoring and control.

Risk Monitoring

Risk monitoring is used to evaluate the effectiveness of risk handling activities against established metrics and provide feedback to the other risk management process steps. Risk monitoring results may also provide a basis to update RHPs, develop additional risk handling options and approaches, and re-analyze risks. In some cases, monitoring results may also be used to identify new risks, revise an existing risk with a new facet, or revise some aspects of risk planning (DAU 2003a, p. 20). Some risk monitoring approaches that can be applied include earned value, program metrics, TPMs, schedule analysis, and variations in risk level. Risk monitoring approaches should be updated and evaluated at the same time and WBS level; otherwise, the results may be inconsistent.

Opportunity and Opportunity Management

In principle, opportunity management is the duality to risk management, with two components: (1) probability of achieving an improved outcome and (2) impact of achieving the outcome. Thus, both should be addressed in risk management planning and execution. In practice, however, a positive opportunity exposure will not match a negative risk exposure in utility space, since the positive utility magnitude of improving an expected outcome is considerably less than the negative utility magnitude of failing to meet an expected outcome (Canada 1971; Kahneman-Tversky 1979). Further, since many opportunity-management initiatives have failed to anticipate serious side effects, all candidate opportunities should be thoroughly evaluated for potential risks to prevent unintended consequences from occurring.

In addition, while opportunities may provide potential benefits for the system or project, each opportunity pursued may have associated risks that detract from the expected benefit. This may reduce the ability to achieve the anticipated effects of the opportunity, in addition to any limitations associated with not pursing an opportunity.

Linkages to Other Systems Engineering Management Topics

The measurement process provides indicators for risk analysis. Project planning involves the identification of risk and planning for stakeholder involvement. Project assessment and control monitors project risks. Decision management evaluates alternatives for selection and handling of identified and analyzed risks.

Practical Considerations

Key pitfalls and good practices related to systems engineering risk management are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in performing risk management are below in Table 1.

Table 1. Risk Management Pitfalls. (SEBoK Original)

Name	Description
Process Over-Reliance	• Over-reliance on the process side of risk management without sufficient attention to human and organizational behavioral considerations.
Lack of Continuity	• Failure to implement risk management as a continuous process. Risk management will be ineffective if it's done just to satisfy project reviews or other discrete criteria. (Charette, Dwinnell, and McGarry 2004, 18-24 and Scheinin 2008).
Tool and Technique Over-Reliance	• Over-reliance on tools and techniques, with insufficient thought and resources expended on how the process will be implemented and run on a day-to-day basis.
Lack of Vigilance	• A comprehensive risk identification will generally not capture all risks; some risks will always escape detection, which reinforces the need for risk identification to be performed continuously.
Automatic Mitigation Selection	• Automatically select the risk handling mitigation option, rather than evaluating all four options in an unbiased fashion and choosing the "best" option.
Sea of Green	• Tracking progress of the risk handling plan, while the plan itself may not adequately include steps to reduce the risk to an acceptable level. Progress indicators may appear "green" (acceptable) associated with the risk handling plan: budgeting, staffing, organizing, data gathering, model preparation, etc. However, the risk itself may be largely unaffected if the handling strategy and the resulting plan is poorly developed, does not address potential root cause(s), and does not incorporate actions that will effectively resolve the risk.
Band-Aid Risk Handling	• Handling risks (e.g., interoperability problems with changes in external systems) by patching each instance, rather than addressing the root cause(s) and reducing the likelihood of future instances.

Good Practices

Some good practices, gathered from the references are below in Table 2.

Table 2. Risk Management Good Practices. (SEBoK Original)

Name	Description
Top Down and Bottom Up	• Risk management should be both “top down” and “bottom up” in order to be effective. The project manager or deputy need to own the process at the top level. But risk management principles should be considered and used by all project personnel.
Early Planning	• Include the planning process step in the risk management process. Failure to adequately perform risk planning early in the project phase, contributes to ineffective risk management.
Risk Analysis Limitations	• Understand the limitations of risk analysis tools and techniques. Risk analysis results should be challenged because considerable input uncertainty and/or potential errors may exist.
Robust Risk Handling Strategy	• The risk handling strategy should attempt to reduce both the probability and consequence of occurrence terms. It is also imperative that the resources needed to properly implement the chosen strategy be available in a timely manner, else the risk handling strategy, and the entire risk management process, will be viewed as a “paper tiger.”
Structured Risk Monitoring	• Risk monitoring should be a structured approach to compare actual vs. anticipated cost, performance, schedule, and risk outcomes associated with implementing the RHP. When ad-hoc or unstructured approaches are used, or when risk level vs. time is the only metric tracked, the resulting risk monitoring usefulness can be greatly reduced.
Update Risk Database	• The risk management database (registry) should be updated throughout the course of the program, striking a balance between excessive resources required and insufficient updates performed. Database updates should occur at both a tailored, regular interval and following major program changes.

References

Works Cited

Boehm, B. 1981. *Software Engineering Economics*. Upper Saddle River, NJ, USA: Prentice Hall.

Boehm, B. 1989. *Software Risk Management*. Los Alamitos, CA; Tokyo, Japan: IEEE Computer Society Press: 115-125.

Canada, J.R. 1971. *Intermediate Economic Analysis for Management and Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Carr, M., S. Konda, I. Monarch, F. Ulrich, and C. Walker. 1993. *Taxonomy-based risk identification*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-93-TR-6.

Charette, R., L. Dwinnell, and J. McGarry. 2004. "Understanding the roots of process performance failure." *CROSSTALK: The Journal of Defense Software Engineering* (August 2004): 18-24.

Clemen, R., and T. Reilly. 2001. *Making hard decisions*. Boston, MA, USA: Duxbury.

Conrow, E. 2003. *Effective Risk Management: Some Keys to Success,* 2nd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics (AIAA).

Conrow, E. 2008. "Risk analysis for space systems." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February, 2008, Los Angeles, CA, USA.

Conrow, E. and P. Shishido. 1997. "Implementing risk management on software intensive projects." IEEE *Software.* 14(3) (May/June 1997): 83-9.

DAU. 2003a. *Risk Management Guide for DoD Acquisition: Fifth Edition,* version 2. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU) Press.

DAU. 2003b. *U.S. Department of Defense extension to: A guide to the project management body of knowledge (PMBOK(R) guide), first edition*. Version 1. 1st ed. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)

Press.

DoD. 2015. *Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs*. Washington, DC, USA: Office of the Deputy Assistant Secretary of Defense for Systems Engineering/Department of Defense.

Evans, M., N. Hastings, and B. Peacock. 2000. *Statistical Distributions,* 3rd ed. New York, NY, USA: Wiley-Interscience.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. 2011. Statistical Distributions, 4th ed. New York, NY, USA.

Gallagher, B., P. Case, R. Creel, S. Kushner, and R. Williams. 2005. *A taxonomy of operational risk*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-2005-TN-036.

Gluch, P. 1994. *A Construct for Describing Software Development Risks*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-94-TR-14.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Kerzner, H. 2009. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling.* 10th ed. Hoboken, NJ, USA: John Wiley & Sons.

Kahneman, D., and A. Tversky. 1979. "Prospect theory: An analysis of decision under risk." *Econometrica.* 47(2) (Mar., 1979): 263-292.

Kumamoto, H. and E. Henley. 1996. *Probabilistic Risk Assessment and Management for Engineers and Scientists,* 2nd ed. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) Press.

Law, A. 2007. *Simulation Modeling and Analysis,* 4th ed. New York, NY, USA: McGraw Hill.

Mun, J. 2010. *Modeling Risk,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

NASA. 2002. *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners,* version 1.1. Washington, DC, USA: Office of Safety and Mission Assurance/National Aeronautics and Space Administration (NASA).

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Scheinin, W. 2008. "Start Early and Often: The Need for Persistent Risk Management in the Early Acquisition Phases." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February 2008, Los Angeles, CA, USA.

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Vose, D. 2000. *Quantitative Risk Analysis,* 2nd ed. New York, NY, USA: John Wiley & Sons.

Willis, H.H., A.R. Morral, T.K. Kelly, and J.J. Medby. 2005. *Estimating Terrorism Risk*. Santa Monica, CA, USA: The RAND Corporation, MG-388.

Primary References

Boehm, B. 1981. *Software Engineering Economics.* Upper Saddle River, NJ, USA:Prentice Hall.

Boehm, B. 1989. *Software Risk Management.* Los Alamitos, CA; Tokyo, Japan: IEEE Computer Society Press, p. 115-125.

Conrow, E.H. 2003. *Effective Risk Management: Some Keys to Success,* 2nd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics (AIAA).

DoD. 2015. Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs. Washington, DC, USA: Office of the Deputy Assistant Secretary of Defense for Systems Engineering/Department of Defense.

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Additional References

Canada, J.R. 1971. *Intermediate Economic Analysis for Management and Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Carr, M., S. Konda, I. Monarch, F. Ulrich, and C. Walker. 1993. *Taxonomy-based risk identification*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-93-TR-6.

Charette, R. 1990. *Application Strategies for Risk Management*. New York, NY, USA: McGraw-Hill.

Charette, R. 1989. *Software Engineering Risk Analysis and Management.* New York, NY, USA: McGraw-Hill (MultiScience Press).

Charette, R., L. Dwinnell, and J. McGarry. 2004. "Understanding the roots of process performance failure." *CROSSTALK: The Journal of Defense Software Engineering* (August 2004): 18-24.

Clemen, R., and T. Reilly. 2001. *Making hard decisions*. Boston, MA, USA: Duxbury.

Conrow, E. 2010. "Space program schedule change probability distributions." Paper presented at American Institute of Aeronautics and Astronautics (AIAA) Space 2010, 1 September 2010, Anaheim, CA, USA.

Conrow, E. 2009. "Tailoring risk management to increase effectiveness on your project." Presentation to the Project Management Institute, Los Angeles Chapter, 16 April, 2009, Los Angeles, CA.

Conrow, E. 2008. "Risk analysis for space systems." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February, 2008, Los Angeles, CA, USA.

Conrow, E. and P. Shishido. 1997. "Implementing risk management on software intensive projects." IEEE *Software.* 14(3) (May/June 1997): 83-9.

DAU. 2003a. *Risk Management Guide for DoD Acquisition: Fifth Edition.* Version 2. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU) Press.

DAU. 2003b. *U.S. Department of Defense extension to: A guide to the project management body of knowledge (PMBOK(R) guide),* 1st ed. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU) Press.

Dorofee, A., J. Walker, C. Alberts, R. Higuera, R. Murphy, and R. Williams (eds). 1996. *Continuous Risk Management Guidebook.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU).

Gallagher, B., P. Case, R. Creel, S. Kushner, and R. Williams. 2005. *A taxonomy of operational risk*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-2005-TN-036.

Gluch, P. 1994. *A Construct for Describing Software Development Risks*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-94-TR-14.

Haimes, Y.Y. 2009. *Risk Modeling, Assessment, and Management*. Hoboken, NJ,USA: John Wiley & Sons, Inc.

Hall, E. 1998. *Managing Risk: Methods for Software Systems Development.* New York, NY, USA: Addison Wesley Professional.

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 4. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2014-001-04.

ISO. 2009. *Risk Management—Principles and Guidelines*. Geneva, Switzerland: International Organization for Standardization (ISO), ISO 31000:2009.

ISO/IEC. 2009. *Risk Management—Risk Assessment Techniques*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 31010:2009.

ISO/IEC/IEEE. 2006. *Systems and Software Engineering - Risk Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 16085.

ISO. 2003. *Space Systems - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 17666:2003.

Jones, C. 1994. *Assessment and Control of Software Risks.* Upper Saddle River, NJ, USA: Prentice-Hall.

Kahneman, D. and A. Tversky. 1979. "Prospect theory: An analysis of decision under risk." *Econometrica.* 47(2) (Mar., 1979): 263-292.

Kerzner, H. 2009. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling,* 10th ed. Hoboken, NJ: John Wiley & Sons.

Kumamoto, H., and E. Henley. 1996. *Probabilistic Risk Assessment and Management for Engineers and Scientists,* 2nd ed. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) Press.

Law, A. 2007. *Simulation Modeling and Analysis,* 4th ed. New York, NY, USA: McGraw Hill.

MITRE. 2012. *Systems Engineering Guide to Risk Management.* Available online: http://www.mitre.org/work/systems_engineering/guide/acquisition_systems_engineering/risk_management/. Accessed on July 7, 2012. Page last updated on May 8, 2012.

Mun, J. 2010. *Modeling Risk,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

NASA. 2002. *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners,* version 1.1. Washington, DC, USA: Office of Safety and Mission Assurance/National Aeronautics and Space Administration (NASA).

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Scheinin, W. 2008. "Start Early and Often: The Need for Persistent Risk Management in the Early Acquisition Phases." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February 2008, Los Angeles, CA, USA.

USAF. 2005. *SMC systems engineering primer & handbook: Concepts, processes, and techniques,* 3rd ed. Los Angeles, CA, USA: Space & Missile Systems Center/U.S. Air Force (USAF).

USAF. 2014. S*MC Risk Management Process Guide*. Version 2. Los Angeles, CA, USA: Space & Missile Systems Center/U.S. Air Force (USAF).

Vose, D. 2000. *Quantitative Risk Analysis.* 2nd ed. New York, NY, USA: John Wiley & Sons.

Willis, H.H., A.R. Morral, T.K. Kelly, and J.J. Medby. 2005. *Estimating Terrorism Risk*. Santa Monica, CA, USA: The RAND Corporation, MG-388.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Measurement

Measurement and the accompanying analysis are fundamental elements of systems engineering (SE) and technical management. SE measurement provides information relating to the products developed, services provided, and processes implemented to support effective management of the processes and to objectively evaluate product or service quality. Measurement supports realistic planning, provides insight into actual performance, and facilitates assessment of suitable actions (Roedler and Jones 2005, 1-65; Frenz et al. 2010).

Appropriate measures and indicators are essential inputs to tradeoff analyses to balance cost, schedule, and technical objectives. Periodic analysis of the relationships between measurement results and review of the requirements and attributes of the system provides insights that help to identify issues early, when they can be resolved with less impact. Historical data, together with project or organizational context information, forms the basis for the predictive models and methods that should be used.

Fundamental Concepts

The discussion of measurement in this article is based on some fundamental concepts. Roedler et al. (2005, 1-65) states three key SE measurement concepts that are paraphrased here:

1. **SE measurement is a consistent but flexible process** that is tailored to the unique information needs and characteristics of a particular project or organization and revised as information needs change.
2. **Decision makers must understand what is being measured.** Key decision-makers must be able to connect *what is being measured* to *what they need to know* and *what decisions they need to make* as part of a closed-loop, feedback control process (Frenz et al. 2010).
3. **Measurement must be used to be effective.**

Measurement Process Overview

The measurement process as presented here consists of four activities from Practical Software and Systems Measurement (PSM) (2011) and described in (ISO/IEC/IEEE 15939; McGarry et al. 2002):

1. establish and sustain commitment
2. plan measurement
3. perform measurement
4. evaluate measurement

This approach has been the basis for establishing a common process across the software and systems engineering communities. This measurement approach has been adopted by the Capability Maturity Model Integration (CMMI) measurement and analysis process area (SEI 2006, 10), as well as by international systems and software engineering standards (ISO/IEC/IEEE 15939; ISO/IEC/IEEE 15288, 1). The International Council on Systems Engineering (INCOSE) Measurement Working Group has also adopted this measurement approach for several of their measurement assets, such as the INCOSE SE Measurement Primer (Frenz et al. 2010) and Technical Measurement Guide (Roedler and Jones 2005). This approach has provided a consistent treatment of measurement that allows the engineering community to communicate more effectively about measurement. The process is illustrated in Figure 1 from Roedler and Jones (2005) and McGarry et al. (2002).

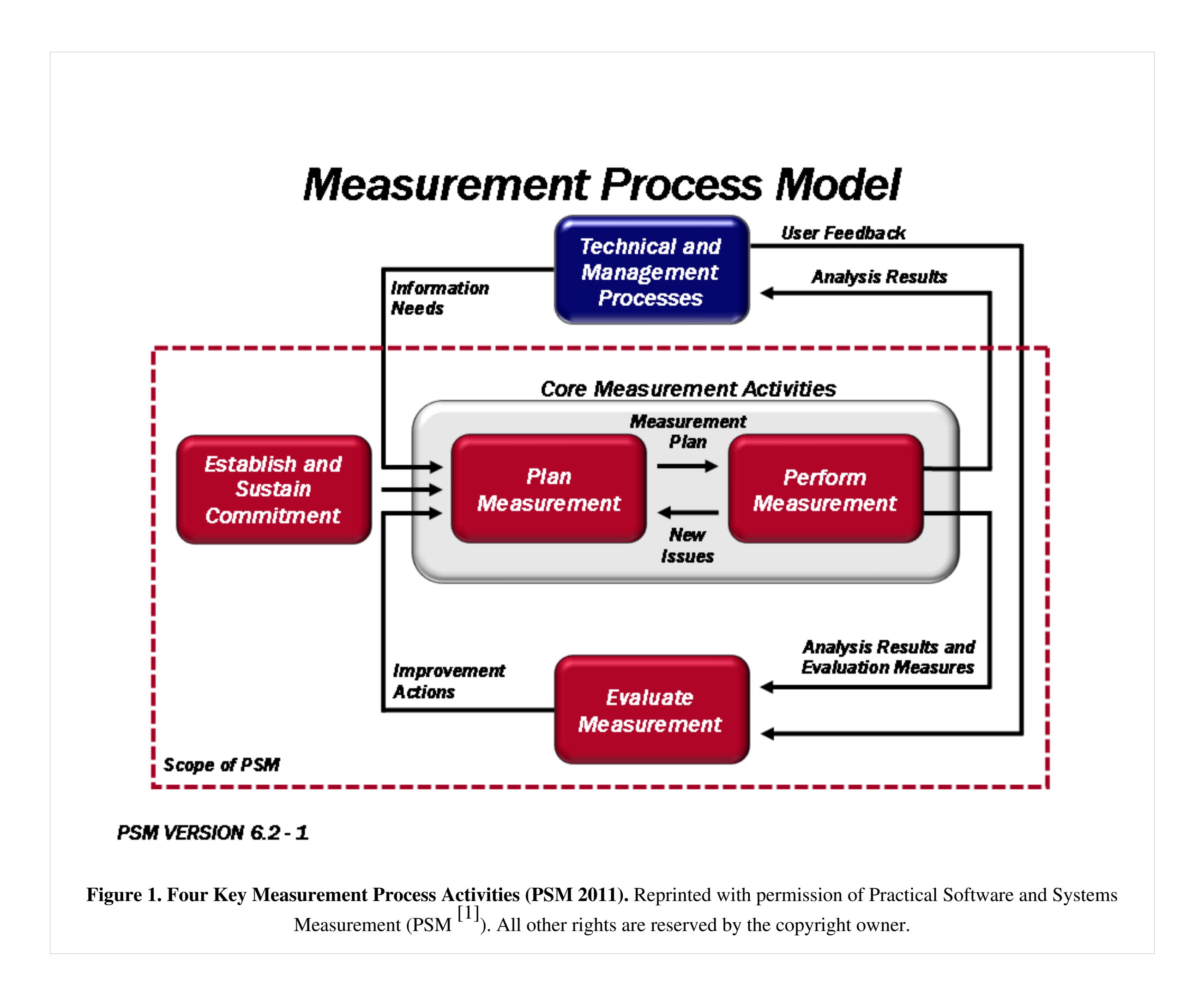


Figure 1. Four Key Measurement Process Activities (PSM 2011). Reprinted with permission of Practical Software and Systems Measurement (PSM [1]). All other rights are reserved by the copyright owner.

Establish and Sustain Commitment

This activity focuses on establishing the resources, training, and tools to implement a measurement process and ensure that there is a management commitment to use the information that is produced. Refer to PSM (August 18, 2011) and SPC (2011) for additional detail.

Plan Measurement

This activity focuses on defining measures that provide insight into project or organization information needs. This includes identifying what the decision-makers need to know and when they need to know it, relaying these information needs to those entities in a manner that can be measured, and identifying, prioritizing, selecting, and specifying measures based on project and organization processes (Jones 2003, 15-19). This activity also identifies the reporting format, forums, and target audience for the information provided by the measures.

Here are a few widely used approaches to identify the information needs and derive associated measures, where each can be focused on identifying measures that are needed for SE management:

- The PSM approach, which uses a set of information categories, measurable concepts, and candidate measures to aid the user in determining relevant information needs and the characteristics of those needs on which to focus (PSM August 18, 2011).
- The (GQM) approach, which identifies explicit measurement goals. Each goal is decomposed into several questions that help in the selection of measures that address the question and provide insight into the goal achievement (Park, Goethert, and Florac 1996).

- Software Productivity Center's (SPC's) 8-step Metrics Program, which also includes stating the goals and defining measures needed to gain insight for achieving the goals (SPC 2011).

The following are good sources for candidate measures that address information needs and measurable concepts/questions:

- PSM Web Site (PSM 2011)
- PSM Guide, Version 4.0, Chapters 3 and 5 (PSM 2000)
- SE Leading Indicators Guide, Version 2.0, Section 3 (Roedler et al. 2010)
- Technical Measurement Guide, Version 1.0, Section 10 (Roedler and Jones 2005, 1-65)
- Safety Measurement (PSM White Paper), Version 3.0, Section 3.4 (Murdoch 2006, 60)
- Security Measurement (PSM White Paper), Version 3.0, Section 7 (Murdoch 2006, 67)
- Measuring Systems Interoperability, Section 5 and Appendix C (Kasunic and Anderson 2004)
- Measurement for Process Improvement (PSM Technical Report), version 1.0, Appendix E (Statz 2005)

The INCOSE *SE Measurement Primer* (Frenz et al. 2010) provides a list of attributes of a good measure with definitions for each attribute; these attributes include *relevance, completeness, timeliness, simplicity, cost effectiveness, repeatability, and accuracy.* Evaluating candidate measures against these attributes can help assure the selection of more effective measures.

The details of each measure need to be unambiguously defined and documented. Templates for the specification of measures and indicators are available on the PSM website (2011) and in Goethert and Siviy (2004).

Perform Measurement

This activity focuses on the collection and preparation of measurement data, measurement analysis, and the presentation of the results to inform decision makers. The preparation of the measurement data includes verification, normalization, and aggregation of the data, as applicable. Analysis includes estimation, feasibility analysis of plans, and performance analysis of actual data against plans.

The quality of the measurement results is dependent on the collection and preparation of valid, accurate, and unbiased data. Data verification, validation, preparation, and analysis techniques are discussed in PSM (2011) and SEI (2010). Per TL 9000, *Quality Management System Guidance*, *The analysis step should integrate quantitative measurement results and other qualitative project information, in order to provide managers the feedback needed for effective decision making* (QuEST Forum 2012, 5-10). This provides richer information that gives the users the broader picture and puts the information in the appropriate context.

There is a significant body of guidance available on good ways to present quantitative information. Edward Tufte has several books focused on the visualization of information, including *The Visual Display of Quantitative Information* (Tufte 2001).

Other resources that contain further information pertaining to understanding and using measurement results include

- PSM (2011)
- ISO/IEC/IEEE 15939, clauses 4.3.3 and 4.3.4
- Roedler and Jones (2005), sections 6.4, 7.2, and 7.3

Evaluate Measurement

This activity involves the analysis of information that explains the periodic evaluation and improvement of the measurement process and specific measures. One objective is to ensure that the measures continue to align with the business goals and information needs, as well as provide useful insight. This activity should also evaluate the SE measurement activities, resources, and infrastructure to make sure it supports the needs of the project and organization. Refer to PSM (2011) and *Practical Software Measurement: Objective Information for Decision Makers* (McGarry et al. 2002) for additional detail.

Systems Engineering Leading Indicators

Leading indicators are aimed at providing predictive insight that pertains to an information need. A SE leading indicator is *a measure for evaluating the effectiveness of a how a specific activity is applied on a project in a manner that provides information about impacts that are likely to affect the system performance objectives* (Roedler et al. 2010). Leading indicators may be individual measures or collections of measures and associated analysis that provide future systems engineering performance insight throughout the life cycle of the system; they *support the effective management of systems engineering by providing visibility into expected project performance and potential future states* (Roedler et al. 2010).

As shown in Figure 2, a leading indicator is composed of characteristics, a condition, and a predicted behavior. The characteristics and conditions are analyzed on a periodic or as-needed basis to predict behavior within a given confidence level and within an accepted time range into the future. More information is also provided by Roedler et al. (2010).

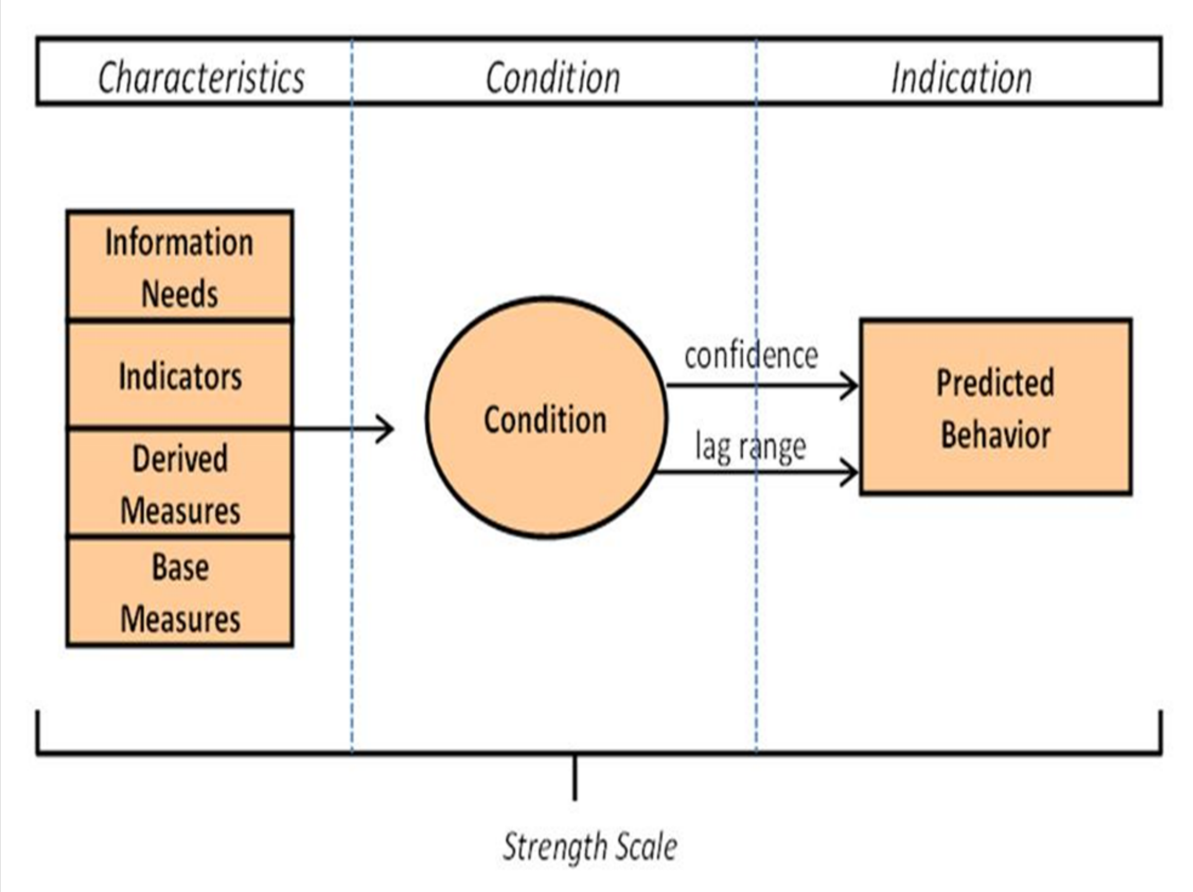


Figure 2. Composition of a Leading Indicator (Roedler et al. 2010). Reprinted with permission of the International Council on Systems Engineering (INCOSE [2]) and Practical Software and Systems Measurement (PSM [1]). All other rights are reserved by the copyright owner.

Technical Measurement

Technical measurement is the set of measurement activities used to provide information about progress in the definition and development of the technical solution, ongoing assessment of the associated risks and issues, and the likelihood of meeting the critical objectives of the acquirer. This insight helps an engineer make better decisions throughout the life cycle of a system and increase the probability of delivering a technical solution that meets both the specified requirements and the mission needs. The insight is also used in trade-off decisions when performance is not within the thresholds or goals.

Technical measurement includes measures of effectiveness (MOEs), measures of performance (MOPs), and technical performance measures (TPMs) (Roedler and Jones 2005, 1-65). The relationships between these types of technical measures are shown in Figure 3 and explained in the reference for Figure 3. Using the measurement process described above, technical measurement can be planned early in the life cycle and then performed throughout the life cycle with increasing levels of fidelity as the technical solution is developed, facilitating predictive insight and preventive or corrective actions. More information about technical measurement can be found in the *NASA Systems Engineering Handbook*, *System Analysis, Design, Development: Concepts, Principles, and Practices*, and the *Systems Engineering Leading Indicators Guide* (NASA December 2007, 1-360, Section 6.7.2.2; Wasson 2006, Chapter 34; Roedler and Jones 2005).

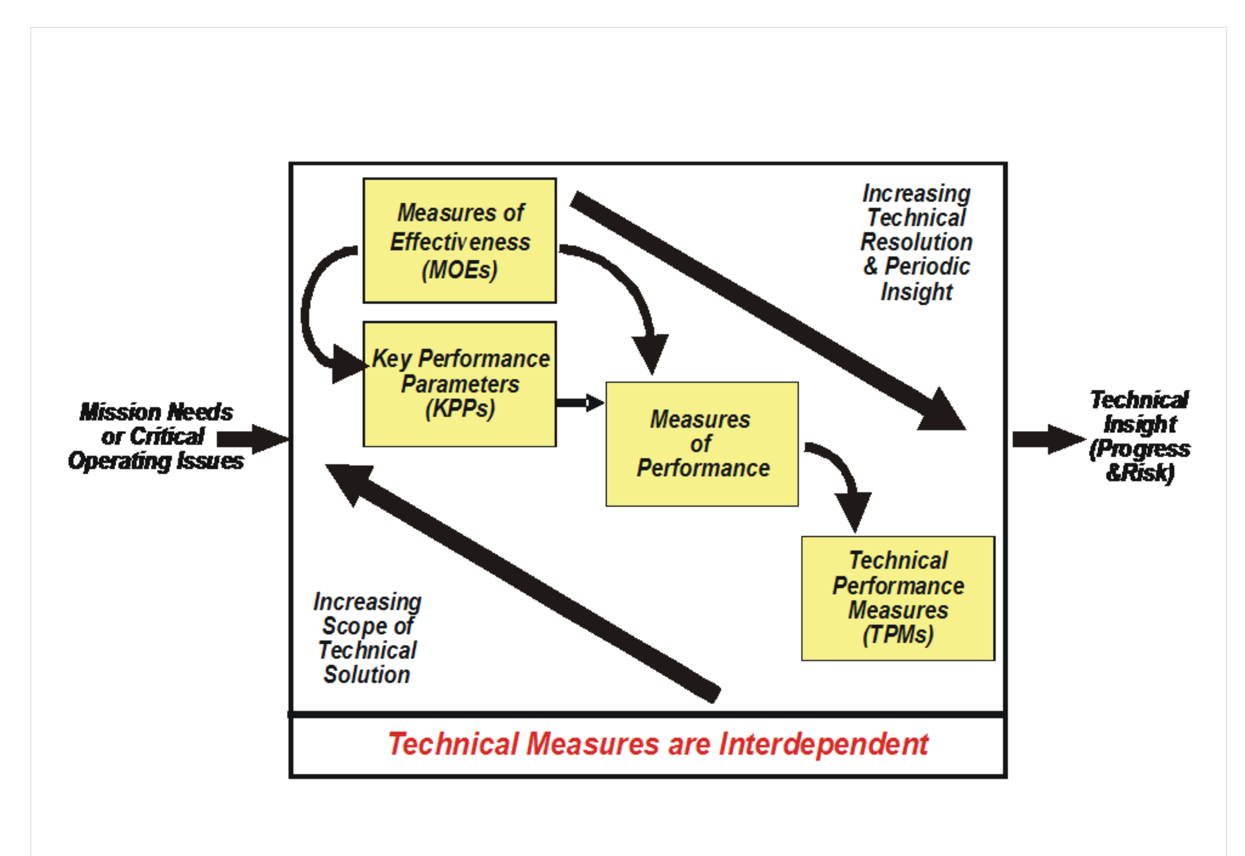


Figure 3. Relationship of the Technical Measures (Roedler et al 2010). Reprinted with permission of the International Council on Systems Engineering (INCOSE [1]) and Practical Software and Systems Measurement (PSM [1]). All other rights are reserved by the copyright owner.

Service Measurement

The same measurement activities can be applied for service measurement; however, the context and measures will be different. Service providers have a need to balance efficiency and effectiveness, which may be opposing objectives. Good service measures are outcome-based, focus on elements important to the customer (e.g., service availability, reliability, performance, etc.), and provide timely, forward-looking information.

For services, the terms critical success factors (CSF) and key performance indicators (KPI) are used often when discussing measurement. CSFs are the key elements of the service or service infrastructure that are most important to achieve the business objectives. KPIs are specific values or characteristics measured to assess achievement of those objectives.

More information about service measurement can be found in the *Service Design* and *Continual Service Improvement* volumes of BMP (2010, 1). More information on service SE can be found in the Service Systems Engineering article.

Linkages to Other Systems Engineering Management Topics

SE measurement has linkages to other SEM topics. The following are a few key linkages adapted from Roedler and Jones (2005):

- Planning – SE measurement provides the historical data and supports the estimation for, and feasibility analysis of, the plans for realistic planning.
- Assessment and Control – SE measurement provides the objective information needed to perform the assessment and determination of appropriate control actions. The use of leading indicators allows for early assessment and control actions that identify risks and/or provide insight to allow early treatment of risks to minimize potential impacts.
- Risk Management – SE risk management identifies the information needs that can impact project and organizational performance. SE measurement data helps to quantify risks and subsequently provides information about whether risks have been successfully managed.
- Decision Management – SE Measurement results inform decision making by providing objective insight.

Practical Considerations

Key pitfalls and good practices related to SE measurement are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing SE Measurement are provided in Table 1.

Table 1. Measurement Pitfalls. (SEBoK Original)

Name	Description
Golden Measures	• Looking for the one measure or small set of measures that applies to all projects. • No one-size-fits-all measure or measurement set exists. • Each project has unique information needs (e.g., objectives, risks, and issues). • The one exception is that, in some cases with consistent product lines, processes, and information needs, a small core set of measures may be defined for use across an organization.
Single-Pass Perspective	• Viewing measurement as a single-pass activity. • To be effective, measurement needs to be performed continuously, including the periodic identification and prioritization of information needs and associated measures.
Unknown Information Need	• Performing measurement activities without the understanding of why the measures are needed and what information they provide. • This can lead to wasted effort.
Inappropriate Usage	• Using measurement inappropriately, such as measuring the performance of individuals or makinng interpretations without context information. • This can lead to bias in the results or incorrect interpretations.

Good Practices

Some good practices, gathered from the references are provided in Table 2.

Table 2. Measurement Good Practices. (SEBoK Original)

Name	Description
Periodic Review	• Regularly review each measure collected.
Action Driven	• Measurement by itself does not control or improve process performance. • Measurement results should be provided to decision makers for appropriate action.
Integration into Project Processes	• SE Measurement should be integrated into the project as part of the ongoing project business rhythm. • Data should be collected as processes are performed, not recreated as an afterthought.
Timely Information	• Information should be obtained early enough to allow necessary action to control or treat risks, adjust tactics and strategies, etc. • When such actions are not successful, measurement results need to help decision-makers determine contingency actions or correct problems.
Relevance to Decision Makers	• Successful measurement requires the communication of meaningful information to the decision-makers. • Results should be presented in the decision-makers preferred format. • Allows accurate and expeditious interpretation of the results.
Data Availability	• Decisions can rarely wait for a complete or perfect set of data, so measurement information often needs to be derived from analysis of the best available data, complemented by real-time events and qualitative insight (including experience).
Historical Data	• Use historical data as the basis of plans, measure what is planned versus what is achieved, archive actual achieved results, and use archived data as a historical basis for the next planning effort.
Information Model	• The information model defined in ISO/IEC/IEEE (2007) provides a means to link the entities that are measured to the associated measures and to the identified information need, and also describes how the measures are converted into indicators that provide insight to decision-makers.

Additional information can be found in the *Systems Engineering Measurement Primer*, Section 4.2 (Frenz et al. 2010), and INCOSE *Systems Engineering Handbook*, Section 5.7.1.5 (2012).

References

Works Cited

Frenz, P., G. Roedler, D.J. Gantzer, P. Baxter. 2010. *Systems Engineering Measurement Primer: A Basic Introduction to Measurement Concepts and Use for Systems Engineering.* Version 2.0. San Diego, CA: International Council on System Engineering (INCOSE). INCOSE-TP-2010-005-02. Accessed April 13, 2015 at http://www.incose.org/ProductsPublications/techpublications/PrimerMeasurement

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2007. *Systems and software engineering - Measurement process*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC/IEEE 15939:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Kasunic, M. and W. Anderson. 2004. *Measuring Systems Interoperability: Challenges and Opportunities.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

McGarry, J., D. Card, C. Jones, B. Layman, E. Clark, J.Dean, F. Hall. 2002. *Practical Software Measurement: Objective Information for Decision Makers*. Boston, MA, USA: Addison-Wesley.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

Park, R.E., W.B. Goethert, and W.A. Florac. 1996. *Goal-Driven Software Measurement – A Guidebook*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU), CMU/SEI-96-BH-002.

PSM. 2011. "Practical Software and Systems Measurement." Accessed August 18, 2011. Available at: http://www.psmsc.com/.

PSM. 2000. *Practical Software and Systems Measurement (PSM) Guide,* version 4.0c. Practical Software and System Measurement Support Center. Available at: http://www.psmsc.com/PSMGuide.asp.

PSM Safety & Security TWG. 2006. *Safety Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SafetyWhitePaper_v3.0.pdf.

PSM Safety & Security TWG. 2006. *Security Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SecurityWhitePaper_v3.0.pdf.

QuEST Forum. 2012. *Quality Management System (QMS) Measurements Handbook,* Release 5.0. Plano, TX, USA: Quest Forum.

Roedler, G., D. Rhodes, C. Jones, and H. Schimmoller. 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2005-001-03.

Roedler, G. and C. Jones. 2005. *Technical Measurement Guide,* version 1.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-020-01.

SEI. 2010. "Measurement and Analysis Process Area" in *Capability Maturity Model Integrated (CMMI) for Development*, version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Software Productivity Center, Inc. 2011. Software Productivity Center web site. August 20, 2011. Available at: http://www.spc.ca/

Statz, J. et al. 2005. *Measurement for Process Improvement,* version 1.0. York, UK: Practical Software and Systems Measurement (PSM).

Tufte, E. 2006. *The Visual Display of Quantitative Information.* Cheshire, CT, USA: Graphics Press.

Wasson, C. 2005. *System Analysis, Design, Development: Concepts, Principles, and Practices*. Hoboken, NJ, USA: John Wiley and Sons.

Primary References

Frenz, P., G. Roedler, D.J. Gantzer, P. Baxter. 2010. *Systems Engineering Measurement Primer: A Basic Introduction to Measurement Concepts and Use for Systems Engineering.* Version 2.0. San Diego, CA: International Council on System Engineering (INCOSE). INCOSE-TP-2010-005-02. Accessed April 13, 2015 at http://www.incose.org/ProductsPublications/techpublications/PrimerMeasurement

ISO/IEC/IEEE. 2007. *Systems and Software Engineering - Measurement Process*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC/IEEE 15939:2007.

PSM. 2000. *Practical Software and Systems Measurement (PSM) Guide,* version 4.0c. Practical Software and System Measurement Support Center. Available at: http://www.psmsc.com.

Roedler, G., D. Rhodes, C. Jones, and H. Schimmoller. 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. San Diego, CA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2005-001-03.

Roedler, G. and C.Jones. 2005. *Technical Measurement Guide,* version 1.0. San Diego, CA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-020-01.

Additional References

Kasunic, M. and W. Anderson. 2004. *Measuring Systems Interoperability: Challenges and Opportunities.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

McGarry, J. et al. 2002. *Practical Software Measurement: Objective Information for Decision Makers*. Boston, MA, USA: Addison-Wesley

NASA. 2007. *NASA Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

Park, Goethert, and Florac. 1996. *Goal-Driven Software Measurement – A Guidebook*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU), CMU/SEI-96-BH-002.

PSM. 2011. "Practical Software and Systems Measurement." Accessed August 18, 2011. Available at: http://www.psmsc.com/.

PSM Safety & Security TWG. 2006. *Safety Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SafetyWhitePaper_v3.0.pdf.

PSM Safety & Security TWG. 2006. *Security Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SecurityWhitePaper_v3.0.pdf.

SEI. 2010. "Measurement and Analysis Process Area" in *Capability Maturity Model Integrated (CMMI) for Development*, version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Software Productivity Center, Inc. 2011. Software Productivity Center web site. August 20, 2011. Available at: http://www.spc.ca/

Statz, J. 2005. *Measurement for Process Improvement,* version 1.0. York, UK: Practical Software and Systems Measurement (PSM).

Tufte, E. 2006. *The Visual Display of Quantitative Information.* Cheshire, CT, USA: Graphics Press.

Wasson, C. 2005. *System Analysis, Design, Development: Concepts, Principles, and Practices*. Hoboken, NJ, USA: John Wiley and Sons.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.psmsc.com
[2] http://www.incose.com

Decision Management

Many systems engineering decisions are difficult because they include numerous stakeholders, multiple competing objectives, substantial uncertainty, and significant consequences. In these cases, good decision making requires a formal decision management process. The purpose of the decision management process is:

> "…to provide a structured, analytical framework for objectively identifying, characterizing and evaluating a set of alternatives for a decision at any point in the life cycle and select the most beneficial course of action."(ISO/IEC/IEEE 15288)

Decision situations (opportunities) are commonly encountered throughout a system's lifecycle. The decision management method most commonly employed by systems engineers is the trade study. Trade studies aim to define, measure, and assess shareholder and stakeholder value to facilitate the decision maker's search for an alternative that represents the best balance of competing objectives. By providing techniques for decomposing a trade decision into logical segments and then synthesizing the parts into a coherent whole, a decision management process allows the decision maker to work within human cognitive limits without oversimplifying the problem. Furthermore, by decomposing the overall decision problem, experts can provide assessments of alternatives in their area of expertise.

Decision Management Process

The decision analysis process is depicted in Figure 1 below. The decision management process is based on several best practices, including:

- Utilizing sound mathematical technique of decision analysis for trade studies. Parnell (2009) provided a list of decision analysis concepts and techniques.
- Developing one master decision model, followed by its refinement, update, and use, as required for trade studies throughout the system life cycle.
- Using Value-Focused Thinking (Keeney 1992) to create better alternatives.
- Identifying uncertainty and assessing risks for each decision.

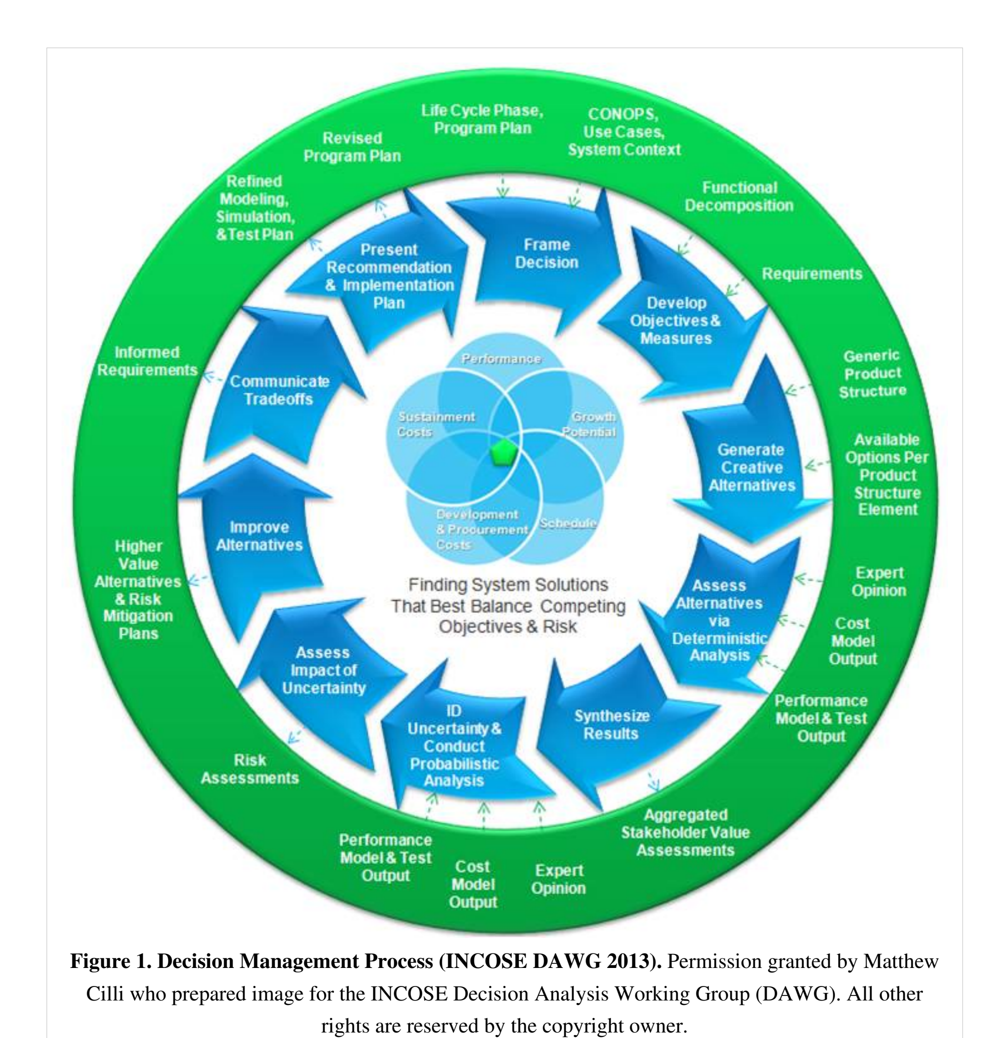


Figure 1. Decision Management Process (INCOSE DAWG 2013). Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

The center of the diagram shows the five trade space objectives (listed clockwise): Performance, Growth Potential, Schedule, Development & Procurement Costs, and Sustainment Costs . The ten blue arrows represent the decision management process activities and the white text within the green ring represents SE process elements. Interactions are represented by the small, dotted green or blue arrows. The decision analysis process is an iterative process. A hypothetical UAV decision problem is used to illustrate each of the activities in the following sections.

Framing and Tailoring the Decision

To ensure the decision team fully understands the decision context, the analyst should describe the system baseline, boundaries and interfaces. The decision context includes: the system definition, the life cycle stage, decision milestones, a list of decision makers and stakeholders, and available resources. The best practice is to identify a decision problem statement that defines the decision in terms of the system life cycle.

Developing Objectives and Measures

Defining how an important decision will be made is difficult. As Keeney (2002) puts it:

> *Most important decisions involve multiple objectives, and usually with multiple-objective decisions, you can't have it all. You will have to accept less achievement in terms of some objectives in order to achieve more on other objectives. But how much less would you accept to achieve how much more?*

The first step is to develop objectives and measures using interviews and focus groups with subject matter experts (SMEs) and stakeholders. For systems engineering trade-off analyses, stakeholder value often includes competing objectives of performance, development schedule, unit cost, support costs, and growth potential. For corporate

decisions, shareholder value would also be added to this list. For performance, a functional decomposition can help generate a thorough set of potential objectives. Test this initial list of fundamental objectives by checking that each fundamental objective is essential and controllable and that the set of objectives is complete, non-redundant, concise, specific, and understandable (Edwards et al. 2007). Figure 2 provides an example of an objectives hierarchy.

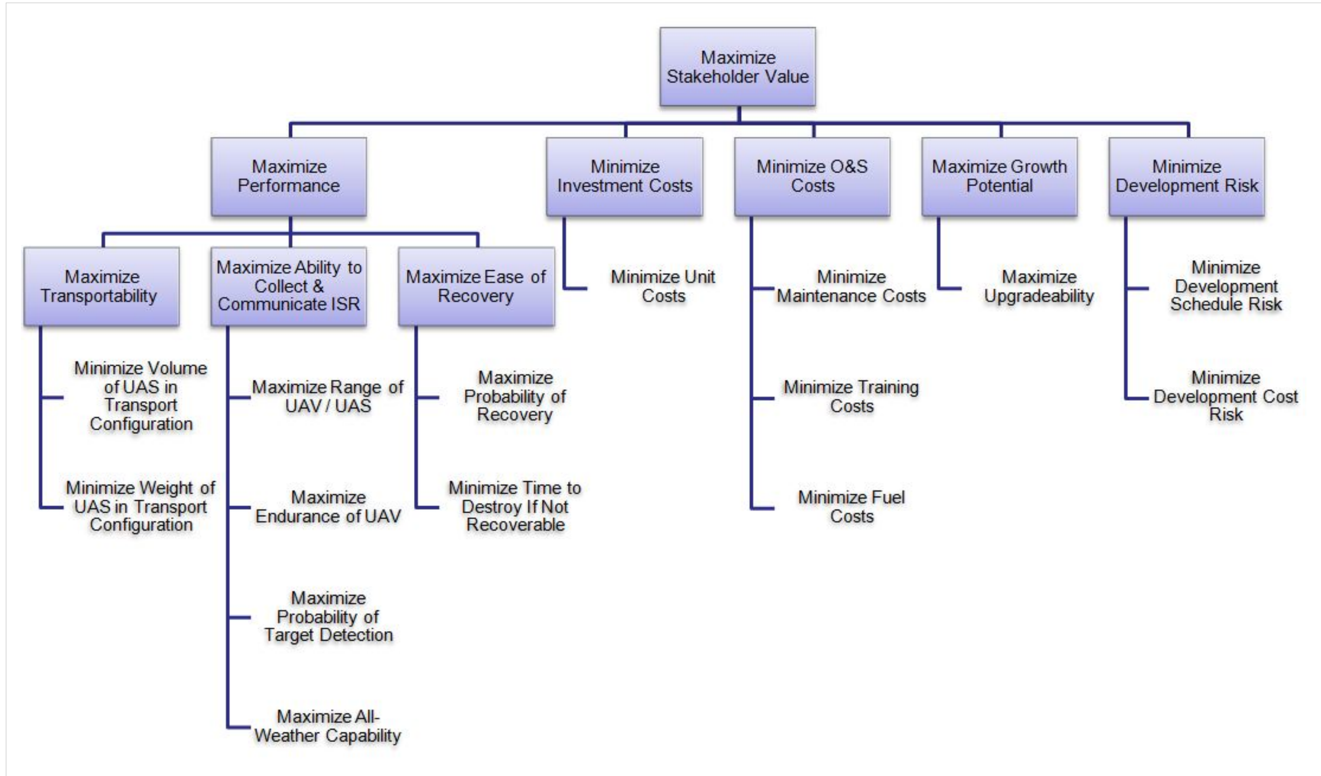


Figure 2. Fundamental Objectives Hierarchy (INCOSE DAWG 2013). Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

For each objective, a measure must be defined to assess the value of each alternative for that objective. A measure (attribute, criterion, and metric) must be unambiguous, comprehensive, direct, operational, and understandable (Keeney & Gregory 2005). A defining feature of multi-objective decision analysis is the transformation from measure space to value space. This transformation is performed by a value function which shows returns to scale on the measure range. When creating a value function, the walk-away point on the measure scale (x-axis) must be ascertained and mapped to a 0 value on the value scale (y-axis). A walk-away point is the measure score where regardless of how well an alternative performs in other measures, the decision maker will walk away from the alternative. He or she does this through working with the user, finding the measure score beyond, at which point an alternative provides no additional value, and labeling it "stretch goal" (ideal) and then mapping it to 100 (or 1 and 10) on the value scale (y-axis). Figure 3 provides the most common value curve shapes. The rationale for the shape of the value functions should be documented for traceability and defensibility (Parnell et al. 2011).

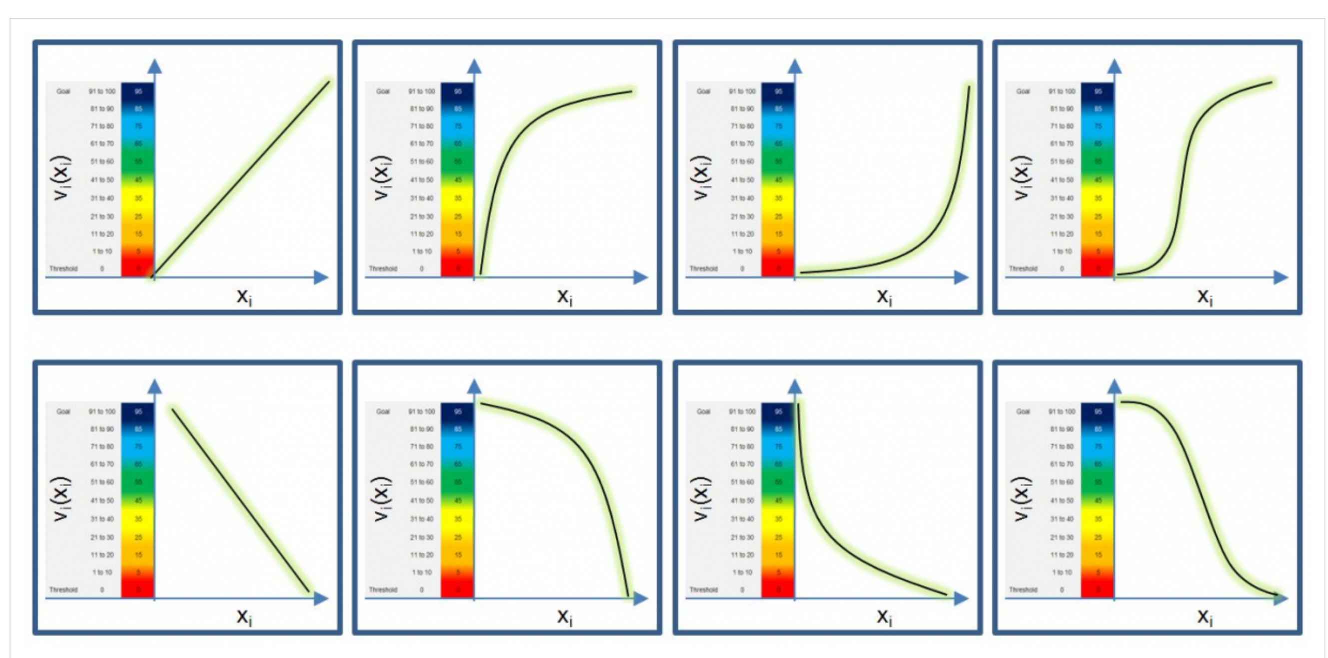


Figure 3. Value Function Examples (INCOSE DAWG 2013). Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG).

The mathematics of multiple objective decision analysis (MODA) requires that the weights depend on importance of the measure and the range of the measure (walk away to stretch goal). A useful tool for determining priority weighting is the swing weight matrix (Parnell et al. 2011). For each measure, consider its importance through determining whether the measure corresponds to a defining, critical, or enabling function and consider the gap between the current capability and the desired capability; finally, put the name of the measure in the appropriate cell of the matrix (Figure 4). The highest priority weighting is placed in the upper-left corner and assigned an unnormalized weight of 100. The unnormalized weights are monotonically decreasing to the right and down the matrix. Swing weights are then assessed by comparing them to the most important value measure or another assessed measure. The swing weights are normalized to sum to one for the additive value model used to calculate value in a subsequent section.

		Level of importance of the value measure								
Highest Weight		Mission Critical	Swt	Mwt	Mission Effectiveness	Swt	Mwt	Mission Enhancing	Swt	Mwt
Capability gap	Large									
		UAV range in nm	100	0.285	All weather capability	50	0.142	UAV P[R]	10	0.028
	Medium									
		UAV endurance in minutes	80	0.228	UAS volume in ft3	40	0.114			
	Small									
		UAV P[D]	50	0.142	UAS weight in pounds	20	0.057	UAV destruct time in seconds	1	0.003

Total swing weight 351

f_i = swing weight

w_i = measure weight

$$w_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

Figure 4. Swing Weight Matrix (INCOSE DAWG 2013). Permission granted by Gregory Parnell who prepared image for the INCOSE Decision Analysis Working Group (DAWG).

Generating Creative Alternatives

To help generate a creative and comprehensive set of alternatives that span the decision space, consider developing an alternative generation table (also called a morphological box) (Buede, 2009; Parnell et al. 2011). It is a best practice to establish a meaningful product structure for the system and to be reported in all decision presentations (Figure 5).

	Cardinal	Buzzard	Crow	Pigeon	Robin	Dove
Subsystem	Design Choice	Design Choice	Design Choice	Design Choice	Design Choice	Design Choice
Propulsion System	Electric 300W & Li P	Electric 300W w/ Li Ion	Electric 600W w/ Solar	Elect 600W w Fuel Cell	Piston Engine 2.5 HP	Piston Engine 4.0 HP
Fuel	NA	NA	NA	NA	JP-8	JP-8
Fuel Tank Capacity	NA	NA	NA	NA	5 liter	7 liter
Propeller	18" Rear	20" rear	22" rear	24" Front	26" Front	28" Front
Wing Configuration	5 ft, Conventional	6 ft, Canard	6 ft, Tandem Wing	7 ft, Three Surface	8 ft., Conventional	9 ft., Conventional
Fin Configuration	Twin Boom Conv.	Inverted V	V Tail	Conventional	H Tail	Cruciform
Actuators	Electromagnetic	Hydraulic	MEMS	Hydraulic	Hydraulic	Hydraulic
Fuselage X Section	12" Diameter	14" Diameter	16" Diameter	18" Diameter	20" Diameter	22" Diameter
Airframe Material	Graphite Epoxy	Graphite Epoxy	Aramid-epoxy	Boron-epoxy	Fiberglass-epoxy	Fiberglass-epoxy
Avionics Arch.	Simplex	Simplex	Triplex	Triplex	Triplex	Triplex
Navigation Sensor	MEMS GPS / INS	MEMS GPS / INS	MEMS GPS / INS	MEMS GPS / INS	MEMS GPS / INS	MEMS GPS / INS
External Comms	LOS COMM Link	LOS COMM Link	LOS + SATCOM Link	LOS + SATCOM Link	LOS + SATCOM Link	LOS + SATCOM Link
Internal Comms	MIL-STD-1553B	MIL-STD-1553B	MIL-STD-1553B	MIL-STD-1553B	MIL-STD-1553B	MIL-STD-1553B
Autopilot	Pre-Programmed, Auto	Semi-Autonomous	Remotely Piloted	Pre-Programmed, Auto	Pre-Programmed, Auto	Pre-Programmed, Auto
Launch / Recovery	Hand / Belly	Hand / Belly	Hand / Belly	Hand / Belly	Hand / Belly	Hand / Belly
Acquisition Sensor	Un-cooled IR	Day Video	Day Video, Cooled IR	Day Video, Cooled IR	Day Video	SAR, Acoustic, Day, IR
Sensor Actuation	Pan-tilt	Pan-tilt-roll	Roll-tilt	Pan-tilt	Pan tilt	Pan tilt
Characteristics	Measurement	Measurement	Measurement	Measurement	Measurement	Measurement
Weight	5 lbs	10 lbs	10 lbs	15 lbs	30 lbs	40 lbs
Max Airspeed	60 kph	50 kph	80 kph	70 kph	60 kph	80 kph
Climb Rate	200 m / minute	150 m / minute	250 m / minute	200 m / minute	200 m / minute	250 m / minute

Figure 5. Descriptions of Alternatives (INCOSE DAWG 2013). Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

Assessing Alternatives via Deterministic Analysis

With objectives and measures established and alternatives having been defined, the decision team should engage SMEs, equipped with operational data, test data, simulations, models, and expert knowledge. Scores are best captured on scoring sheets for each alternative/measure combination which document the source and rationale. Figure 6 provides a summary of the scores.

Raw Scorecard (expected pefromance values)			RELOCATE UAV		EMPLOY UAV				RECOVER UAV		GROWTH POT.	UNIT COST	DEVELOPMENT RISK		OPERATION & SUPPORT COST		
			Minimize UAV weight	Minimize UAV volume	Maximize all weather capability	Maximize UAV range	Maximize UAV probability of detection	Maximize UAV endurance	Maximize probability of recovery	Minimize time to destroy if not recoverable	Maximize upgradability	Minimize unit cost	Minimize development schedule risk	Minimize development cost risk	Minimize training cost	Minimize maintenance cost	Minimize fuel cost
ID	Name	Image	(lbs)	ft3	index	km	P[D]	hours	P[R]	sec	index	FY13$K	index	index	FY13$	FY13$	FY13$
1	Cardinal		5	12	3	10	0.92	0.5	0.6	1	0.3	250	0.9	0.9	300	300	0
2	Buzzard		10	15	1	10	0.9	1	0.7	2	0.6	300	0.8	0.8	300	300	0
3	Crow		10	20	3	70	0.92	1	0.8	2	0.6	350	0.7	0.7	300	300	0
4	Pigeon		15	30	3	80	0.92	1.5	0.9	2	0.6	400	0.6	0.6	300	300	0
5	Robin		30	40	1	90	0.9	2	0.9	2	0.6	500	0.5	0.5	500	500	300
6	Dove		40	50	5	100	0.94	2	0.9	3	0.9	700	0.4	0.4	500	500	500
7	Ideal		5	10	5	100	1	2	0.9	1	1	200	1	1	250	0	0

Figure 6. Alternative Scores (INCOSE DAWG 2013). Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG).

Note that in addition to identified alternatives, the score matrix includes a row for the ideal alternative. The ideal is a tool for value-focused thinking, which will be covered later.

Synthesizing Results

Next, one can transform the scores into a value table, by using the value functions developed previously. A color heat map can be useful to visualize value tradeoffs between alternatives and identify where alternatives need improvement (Figure 7).

		Heat-indexed Value Scorecard	RELOCATE UAV		EMPLOY UAV				RECOVER UAV		GROWTH POT.	UNIT COST	DEVELOPMENT RISK		OPERATION & SUPPORT COST		
			Minimize UAV weight	Minimize UAV volume	Maximize all weather capability	Maximize UAV range	Maximize UAV probability of detection	Maximize UAV endurance	Maximize probability of recovery	Minimize time to destroy if not recoverable	Maximize upgradability	Minimize unit cost	Minimize development schedule risk	Minimize development cost risk	Minimize training cost	Minimize maintenance cost	Minimize fuel cost
ID	Name	Image	0.06	0.11	0.14	0.28	0.14	0.23	0.03	0.01	1	1	0.5	0.5	0.33	0.33	0.33
1	Cardinal		100	97	90	13	59	1	47	100	31	94	83	83	83	85	100
2	Buzzard		86	92	1	13	42	60	77	90	61	88	67	67	83	85	100
3	Crow		86	85	90	83	59	60	90	75	61	82	50	50	83	85	100
4	Pigeon		72	57	90	90	59	80	100	75	61	76	34	34	83	85	100
5	Robin		29	29	1	95	42	100	100	60	61	56	18	18	17	67	51
6	Dove		1	1	100	100	75	100	100	1	91	1	1	1	17	67	18
7	Ideal		100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

Figure 7. Value Scorecard with Heat Map (INCOSE DAWG 2013). Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

The additive value model uses the following equation to calculate each alternative's value:

$$v(x) = \sum_{i=1}^{n} w_i v_i(x_i)$$

where

$v(x)$ is the alternative's value

$i = 1 \; to \; n$ is the number of the measure

x_i is the alternative's score on the ith measure

$v_i(x_i)$ is the single dimensional value of a score of x_i

w_i is the weight of the ith measure

and $\sum_{i=1}^{n} w_i = 1$ (all weights sum to one)

The value component chart (Figure 8) shows the total value and the weighted value measure contribution of each alternative (Parnell et al. 2011).

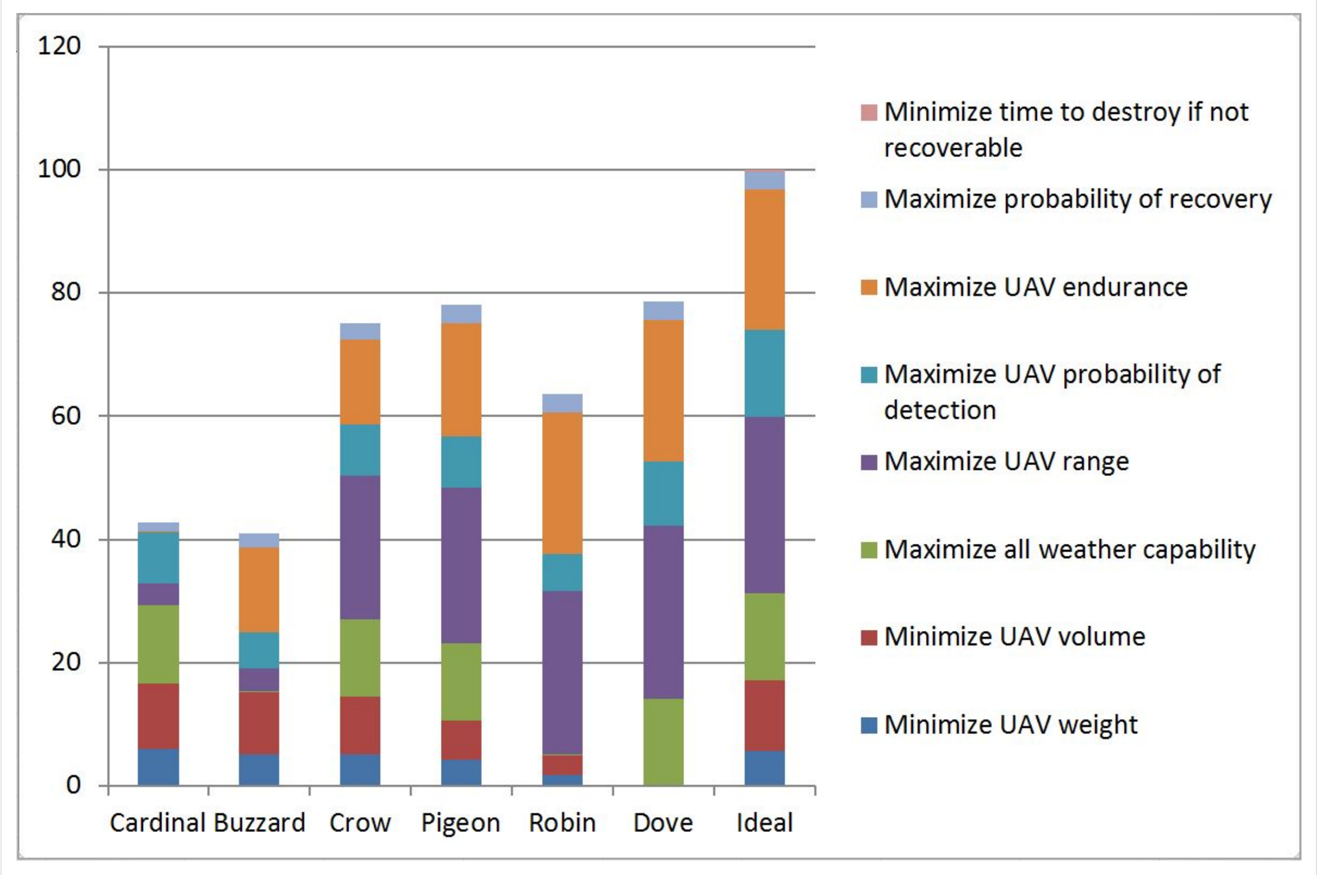


Figure 8. Value Component Graph (INCOSE DAWG 2013). Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

The heart of a decision management process for system engineering trade off analysis is the ability to assess all dimensions of shareholder and stakeholder value. The stakeholder value scatter plot in Figure 9 shows five dimensions: unit cost, performance, development risk, growth potential, and operation and support costs for all alternatives.

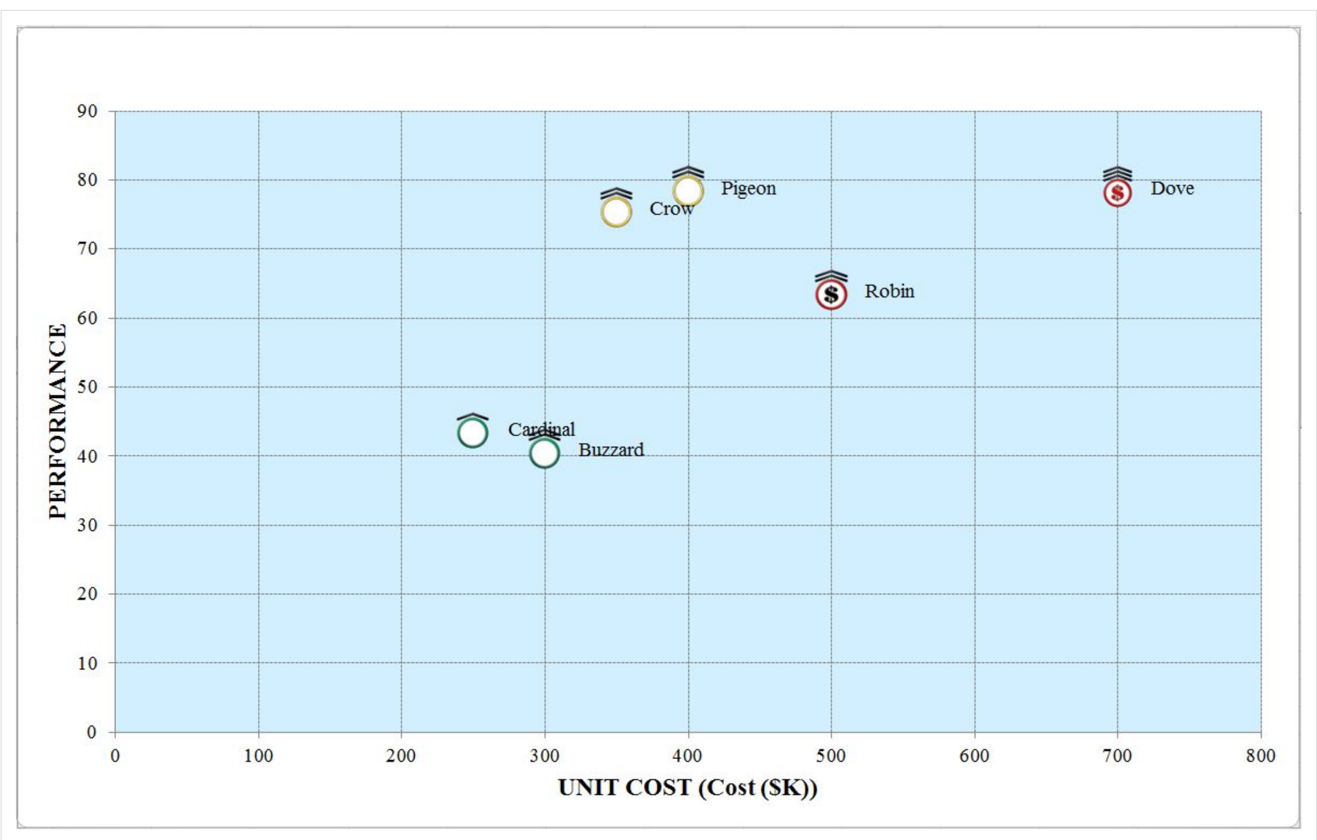


Figure 9. Example of a Stakeholder Value Scatterplot (INCOSE DAWG 2013). Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

Each system alternative is represented by a scatter plot marker (Figure 9). An alternative's unit cost and performance value are indicated by x and y positions respectively. An alternative's development risk is indicated by the color of the marker (green = low, yellow= medium, red = high), while the growth potential is shown as the number of hats above the circular marker (1 hat = low, 2 hats = moderate, 3 hats = high).

Identifying Uncertainty and Conducting Probabilistic Analysis

As part of the assessment, the SME should discuss the potential uncertainty of the independent variables. The independent variables are the variables that impact one or more scores; the scores that are independent scores. Many times the SME can assess an upper, nominal, and lower bound by assuming low, moderate, and high performance. Using this data, a Monte Carlo Simulation summarizes the impact of the uncertainties and can identify the uncertainties that have the most impact on the decision.

Accessing Impact of Uncertainty - Analyzing Risk and Sensitivity

Decision analysis uses many forms of sensitivity analysis including line diagrams, tornado diagrams, waterfall diagrams and several uncertainty analyses including Monte Carlo Simulation, decision trees, and influence diagrams (Parnell et al. 2013). A line diagram is used to show the sensitivity to the swing weight judgment (Parnell et al. 2011). Figure 10 shows the results of a Monte Carlo Simulation of performance value.

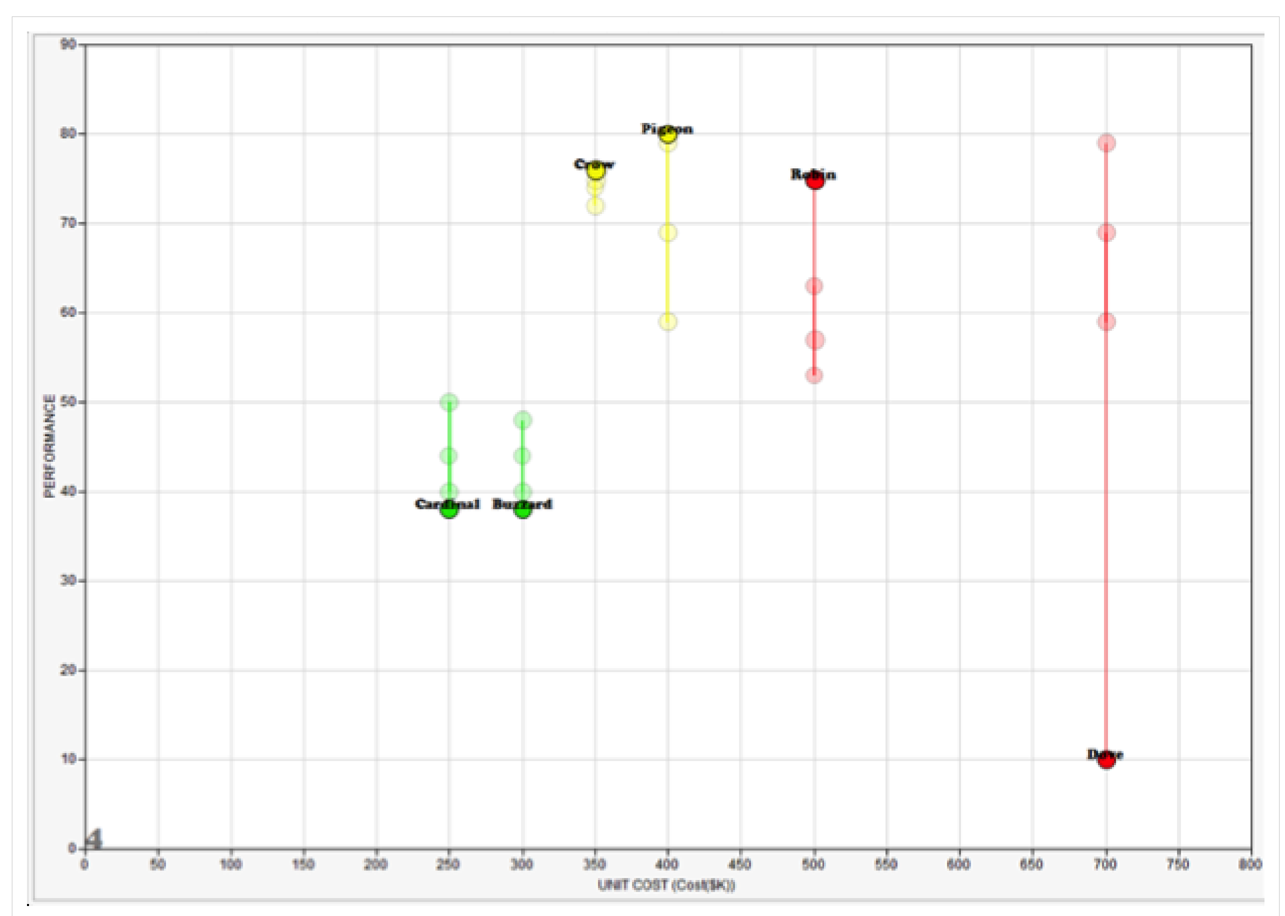


Figure 10. Uncertainty on Performance Value from Monte Carlo Simulation (INCOSE DAWG 2013). Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

Improving Alternatives

Mining the data generated for the alternatives will likely reveal opportunities to modify some design choices to claim untapped value and/or reduce risk. Taking advantage of initial findings to generate new and creative alternatives starts the process of transforming the decision process from "alternative-focused thinking" to "value-focused thinking" (Keeney 1993).

Communicating Tradeoffs

This is the point in the process where the decision analysis team identifies key observations about tradeoffs and the important uncertainties and risks.

Presenting Recommendations and Implementing Action Plan

It is often helpful to describe the recommendation(s) in the form of a clearly-worded, actionable task-list in order to increase the likelihood of the decision implementation. Reports are important for historical traceability and future decisions. Take the time and effort to create a comprehensive, high-quality report detailing study findings and supporting rationale. Consider static paper reports augmented with dynamic hyper-linked e-reports.

References

Works Cited

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ: John Wiley & Sons Inc.

Edwards, W., R.F. Miles Jr., and D. Von Winterfeldt. 2007. *Advances In Decision Analysis: From Foundations to Applications.* New York, NY: Cambridge University Press.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Keeney, R.L. and H. Raiffa H. 1976. *Decisions with Multiple Objectives - Preferences and Value Tradeoffs.* New York, NY: Wiley.

Keeney, R.L. 1992. *Value-Focused Thinking: A Path to Creative Decision-Making.* Cambridge, MA: Harvard University Press.

Keeney, R.L. 1993. "Creativity in MS/OR: Value-focused thinking—Creativity directed toward decision making." *Interfaces*, 23(3), p.62–67.

Parnell, G.S. 2009. "Decision Analysis in One Chart," *Decision Line, Newsletter of the Decision Sciences Institute*. May 2009.

Parnell, G.S., P.J. Driscoll, and D.L Henderson (eds). 2011. *Decision Making for Systems Engineering and Management*, 2nd ed. Wiley Series in Systems Engineering. Hoboken, NJ: Wiley & Sons Inc.

Parnell, G.S., T. Bresnick, S. Tani, and E. Johnson. 2013. *Handbook of Decision Analysis.* Hoboken, NJ: Wiley & Sons.

Primary References

Buede, D.M. 2004. "On Trade Studies." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June, 2004, Toulouse, France.

Keeney, R.L. 2004. "Making Better Decision Makers." *Decision Analysis*, 1(4), pp.193–204.

Keeney, R.L. & R.S. Gregory. 2005. "Selecting Attributes to Measure the Achievement of Objectives". *Operations Research*, 53(1), pp.1–11.

Kirkwood, C.W. 1996. *Strategic Decision Making: Multiobjective Decision Analysis with Spreadsheets.* Belmont, California: Duxbury Press.

Additional References

Buede, D.M. and R.W. Choisser. 1992. "Providing an Analytic Structure for Key System Design Choices." *Journal of Multi-Criteria Decision Analysis*, 1(1), pp.17–27.

Felix, A. 2004. "Standard Approach to Trade Studies." Proceedings of the International Council on Systems Engineering (INCOSE) Mid-Atlantic Regional Conference, November 2-4 2004, Arlington, VA.

Felix, A. 2005. "How the Pro-Active Program (Project) Manager Uses a Systems Engineer's Trade Study as a Management Tool, and not just a Decision Making Process." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY.

Miller, G.A. 1956. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." *Psychological Review*, 63(2), p.81.

Ross, A.M. and D.E. Hastings. 2005. "Tradespace Exploration Paradigm." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY.

Sproles, N. 2002. "Formulating Measures of Effectiveness." *Systems Engineering", 5(4), p. 253-263.*

Silletto, H. 2005. "Some Really Useful Principles: A new look at the scope and boundaries of systems engineering." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY.

Ullman, D.G. and B.P. Spiegel. 2006. "Trade Studies with Uncertain Information." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2006, Orlando, FL.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Configuration Management

The purpose of configuration management (CM) is to establish and maintain the integrity of all of the identified outputs of a project or process and make them available to concerned parties (ISO/IEC/IEEE 2015). Unmanaged changes to system artifacts (such as those associated with plans, requirements, design, software, hardware, testing, and documentation) can lead to problems that persist throughout the system life cycle. Hence, one primary objective of CM is to manage and control the change to such artifacts.

Configuration Management Process Overview

CM is the discipline of identifying and formalizing the functional and physical characteristics of a configuration item at discrete points in the product evolution for the purpose of maintaining the integrity of the product system and controlling changes to the baseline. The baseline for a project contains all of the technical requirements and related cost and schedule requirements that are sufficiently mature to be accepted and placed under change control by the project manager. The project baseline consists of two parts: the technical baseline and the business baseline. The systems engineer is responsible for managing the technical baseline and ensuring that it is consistent with the costs and schedules in the business baseline. Typically, the project control office manages the business baseline.

The ANSI/GEIA EIA-649-A standard presents CM from the viewpoint that configuration management practices are employed because they make good business sense rather than because requirements are imposed by an external customer (ANSI/GEIA 2005). The standard discusses CM principles and practices from an enterprise view; it does not prescribe which CM activities individual organizations or teams within the enterprise should perform. Each enterprise assigns responsibilities in accordance with its own management policy. See also the Implementation Guide for Configuration Management, which supports and provides further information on this standard

(ANSI/GEIA October 2005).

Effective CM depends on the establishment, maintenance, and implementation of an effective process. The CM process should include, but are not limited to, the following activities:

- identification and involvement of relevant stakeholders
- setting of CM goals and expected outcomes
- identification and description of CM tasks
- assignment of responsibility and authority for performing the CM process tasks
- establishment of procedures for monitoring and control of the CM process
- measurement and assessment of the CM process effectiveness

As a minimum the CM process should incorporate and detail the following tasks (SEI 2010):

- identifying the configuration of selected work products that compose the baselines at given points in time
- controlling changes to configuration items
- building or providing specifications to build work products from the configuration management system
- maintaining the integrity of baselines
- providing accurate status and current configuration data to developers, end users, and customers

Figure 1 below shows the primary functions of systems CM.



Figure 1. Configuration Management Functions. (SEBoK Original)

Planning

The CM plan must be developed in consideration of the organizational context and culture; it must adhere to or incorporate applicable policies, procedures, and standards and it must accommodate acquisition and subcontractor situations. A CM plan details and schedules the tasks to be performed as part of the CM process including: configuration identification, change control, configuration status accounting, configuration auditing, and release management and delivery.

Configuration Identification

This activity is focused on identifying the configuration items which will be managed and controlled under a CM process. The identification activity involves establishing a procedure for labeling items and their versions. The labeling provides a context for each item within the system configuration and shows the relationship between system items.

Establishing Baseline

Configuration items are typically assembled into a baseline which specifies how a system will be viewed for the purposes of management, control, and evaluation. This baseline is fixed at a specific point in time in the system life cycle and represents the current approved configuration. It generally can only be changed through formal change procedures.

Change Control

A disciplined change control process is critical for systems engineering. A generalized change control process in response to an engineering change proposal (ECP) is shown in Figure 2 below, which is adapted from Systems Engineering and Analysis (Blanchard and Fabrycky 1999).

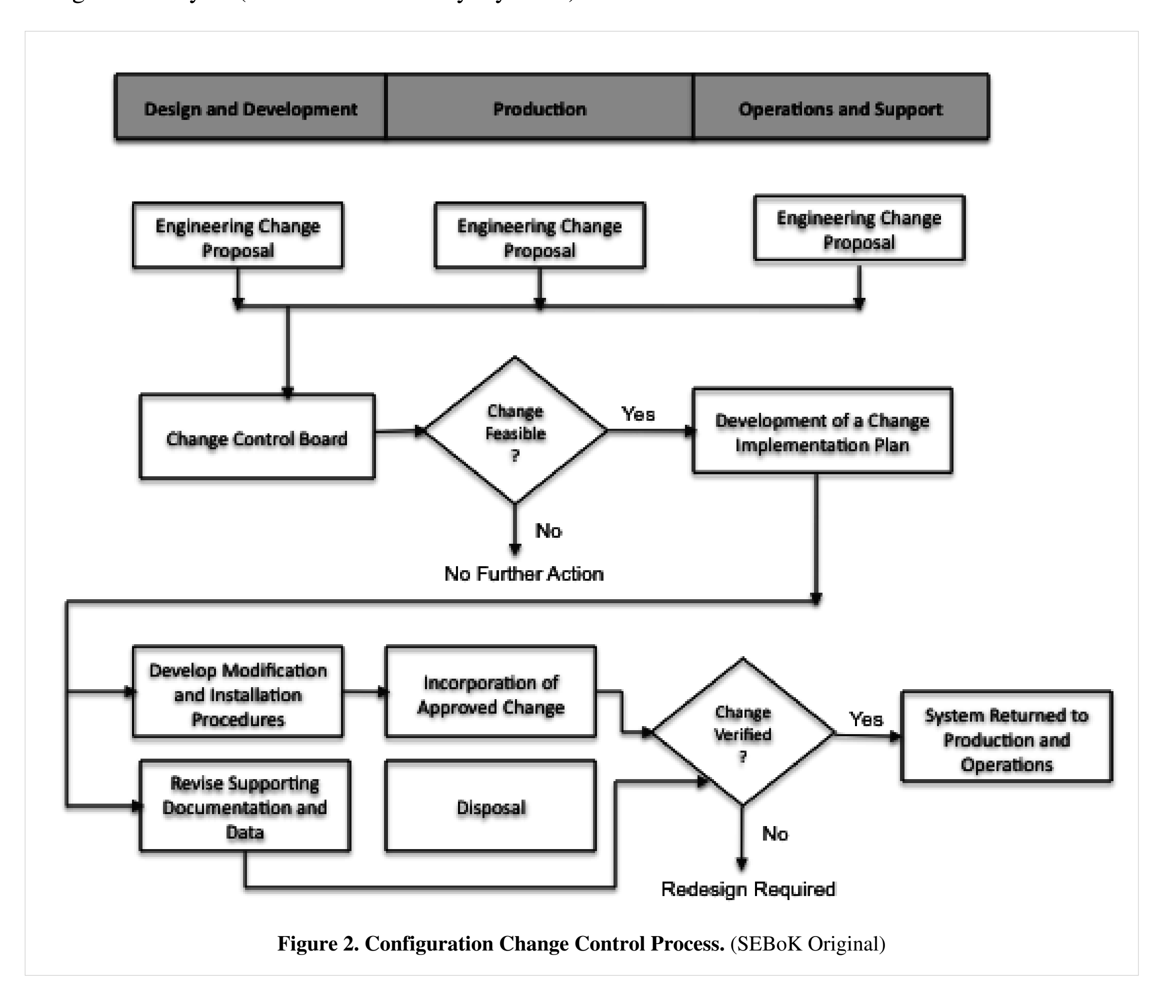


Figure 2. Configuration Change Control Process. (SEBoK Original)

Configuration Auditing

Audits are independent evaluations of the current status of configuration items and determine conformance of the configuration activities to the CM process. Adherence to applicable CM plans, regulations, and standards, is typically assessed during audits.

Constraints and Guidance

Constraints affecting and guiding the CM process come from a number of sources. Policies, procedures, and standards set forth at corporate or other organizational levels might influence or constrain the design and implementation of the CM process. Also, the contract with an acquirer or supplier may contain provisions affecting the CM process. The system life cycle process adopted and the tools, methods, and other processes used in system development can affect the CM process (Bourque and Fairley 2014). There are a variety of sources for guidance on the development of a CM process. These include the ISO standards on system life cycle processes (ISO/IEC/IEEE 15288 2015) and configuration management guidelines (ISO 10007 2003), as well as the *Guide to The Software Engineering Body of Knowledge* (SWEBOK) (Bourque and Fairley 2014), and the CMMI for Development (SEI 2010).

Organizational Issues

Successful CM planning, management, and implementation requires an understanding of the organizational context for on the design and implementation of the CM process and why constraints are placed upon it. To plan a CM process for a project, it is necessary to understand the organizational context and the relationships among the organizational elements. CM interacts with other organizational elements, which may be structured in a number of ways. Although the responsibility for performing certain CM tasks might be assigned to other parts of the organization, the overall responsibility for CM often rests with a distinct organizational element or designated individual (Bourque and Fairley 2014).

Measurement

In order to carry out certain CM functions, such as status accounting and auditing, as well as to monitor and assess the effectiveness of CM processes, it is necessary to measure and collect data related to CM activities and system artifacts. CM libraries and automated report tools provide convenient access and facilitation of data collection. Examples of metrics include the size of documentation artifacts, number of change requests, mean time to change to a configuration item, and rework costs.

Tools

CM employs a variety of tools to support the process, for example:

- library management
- tracking and change management
- version management
- release management

The INCOSE Tools Database Working Group (INCOSE TDWG 2010) maintains an extensive list of tools including configuration management.

Linkages to Other Systems Engineering Management Topics

Configuration management is involved in the management and control of artifacts produced and modified throughout the system life cycle in all areas of system definition, system realization, system deployment and use, and product and service life management. This includes CM application to the artifacts of all the other management processes (plans, analyses, reports, statuses, etc.).

Practical Considerations

Key pitfalls and good practices related to systems engineering CM are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing CM are in Table 1.

Table 1. Configuration Management Pitfalls. (SEBoK Original)

Name	Description
Shallow Visibility	• Not involving all affected disciplines in the change control process.
Poor Tailoring	• Inadequate CM tailoring to adapt to the project scale, number of subsystems, etc.
Limited CM Perspective	• Not considering and integrating the CM processes of all contributing organizations including COTS vendors and subcontractors.

Good Practices

Some good practices gathered from the references are provided in Table 2 below.

Table 2. Configuration Management Good Practices. (SEBoK Original)

Name	Description
Cross-Functional CM	• Implement cross-functional communication and CM processes for software, hardware, firmware, data, or other types of items as appropriate.
Full Lifecycle Perspective	• Plan for integrated CM through the life cycle. Do not assume that it will just happen as part of the program.
CM Planning	• Processes are documented in a single, comprehensive CM plan early in the project. The plan should be a (systems) CM plan. • Include tools selected and used.
Requirements Traceability	• Initiate requirements traceability at the start of the CM activity.
CCB Hierarchy	• Use a hierarchy of configuration control boards commensurate with the program elements.
Consistent Identification	• Software CI and hardware CI use consistent identification schemes.
CM Automation	• Configuration status accounting should be as automated as possible.

Additional good practices can be found in ISO/IEC/IEEE (2009, Clause 6.4) and INCOSE (2010, sec. 5.4.1.5).

References

Works Cited

ANSI/GEIA. 2005. Implementation Guide for Configuration Management. Arlington, VA, USA: American National Standards Institute/Government Electronics & Information Technology Association, GEIA-HB-649. October 2005.

Blanchard, B.S. and W J. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th ed. Prentice-hall international series in industrial and systems engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK).* Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

ISO. 2003. *Quality Management Systems – Guidelines for Configuration Management*. Geneva, Switzerland: International Organization for Standardization (ISO), ISO 10007:2003.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering-- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions.ISO/IEC/IEEE 15288:2015

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Primary References

ANSI/GEIA. 2005. *Implementation Guide for Configuration Management*. Arlington, VA, USA: American National Standards Institute/Government Electronics & Information Technology Association, GEIA-HB-649. October 2005.

GEIA. 2004. *GEIA Consensus Standard for Data Management*. Arlington, VA, USA: Government Electronics & Information Technology Association, GEIA-859.

ISO. 2003. *Quality Management Systems – Guidelines for Configuration Management*. Geneva, Switzerland: International Organization for Standardization (ISO), ISO 10007:2003.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering-- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions.ISO/IEC/IEEE 15288:2015

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Additional References

INCOSE Tools database working group (TDWG). in International Council on Systems Engineering (INCOSE) [database online]. San Diego, CA, USA, 2010. Accessed April 13, 2015 Available at: http://www.incose.org/docs/default-source/wgcharters/tools-database.pdf

INCOSE. 2008. "INCOSE measurement tools survey." in International Council on Systems Engineering (INCOSE) [database online]. San Diego, CA, USA, 2008

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Information Management

The information management (IM) process is a set of activities associated with the collection and management of information from one or more sources and the distribution of that information to one or more audiences. Information, in its most restricted technical sense, is an ordered sequence of symbols that record or transmit a message. The key idea is that information is a collection of facts that is organized in such a way that they have additional value beyond the value of the facts themselves. The systems engineer is both the generator and recipient of information products; thus, the systems engineer has a vital stake in the success of the development and use of the IM process and IM systems.

Overview

Information can exist in many forms in an organization; some information is related to a specific system development program and some is held at an enterprise level and made available to programs as required. It may be held in electronic format or in physical form (for instance, paper drawings or documents, microfiche or other photographic records).

The IM process includes a set of interrelated activities associated with information systems, system of systems (SoS), architectures, services, nested hardware/platforms, and people. Fundamentally, this process is a set of activities that are concerned with improvements in a variety of human problem-solving endeavors. This includes the design, development, and use of technologically-based systems and processes that enhance the efficiency and effectiveness of information and associated knowledge in a variety of strategic/business, tactical, and operational situations.

Management refers to the organization of and control over process activities associated with the structure, processing, and delivery of information. For example, the organizational structure must have management processes capable of managing this information throughout the information life cycle regardless of source or format (e.g., data, paper documents, electronic documents, audio, video, etc.) for delivery through multiple channels that may include cell phones and web interfaces.

A computer-based IM system is an organized combination of people, hardware, software, communication networks, and the data resources that collect, transform, and disseminate information in an organization. From the perspective of the systems engineer, the IM process is a cycle of inter-related information activities to be planned for, designed, and coordinated. Numerous life cycle development process models exist. Figure 1 below is a high level process model that emphasizes the role of systems engineering (SE) in IM.

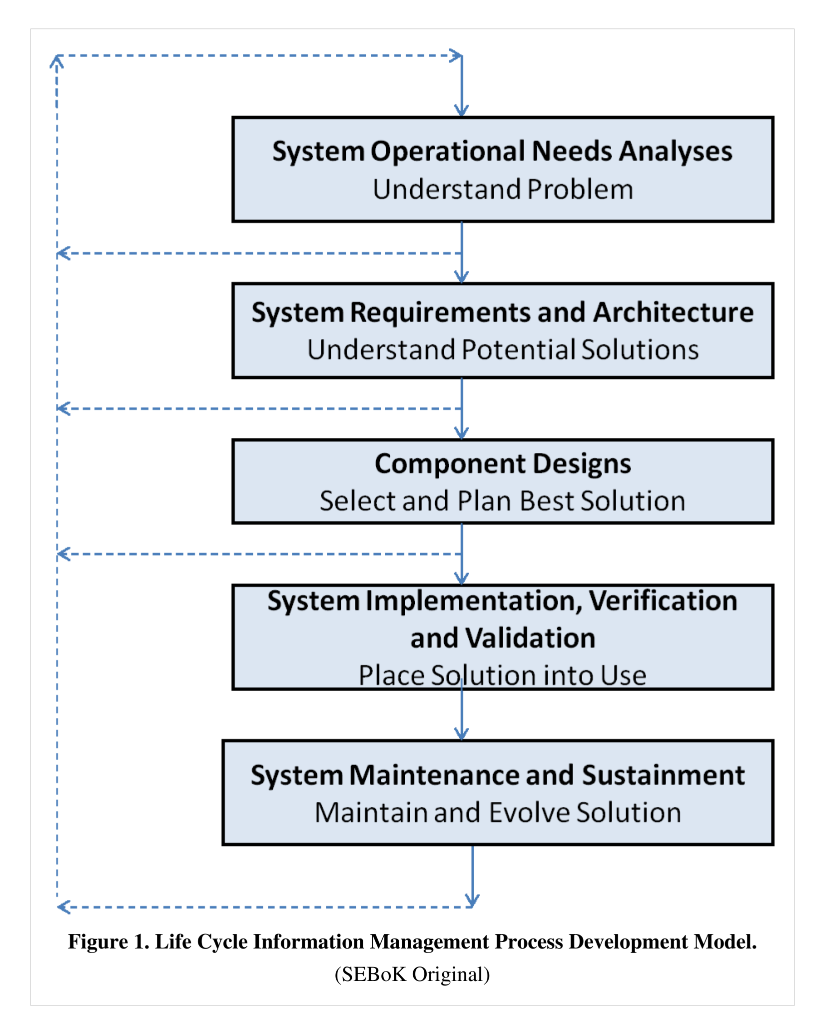


Figure 1. Life Cycle Information Management Process Development Model. (SEBoK Original)

The SE function in the development of an IM system is concerned with several rate-limiting architecture and design variables, (e.g., information sharing, quality, security, efficiency, compliance, etc.) that should be considered up-front in the life cycle development process. Each of these variables can be subdivided into architecture and design considerations for the information system-of-interest (SoI). For example, quality can be viewed in terms of data validity, consistency, and comprehensiveness. Figure 2 provides an overview of information management considerations.

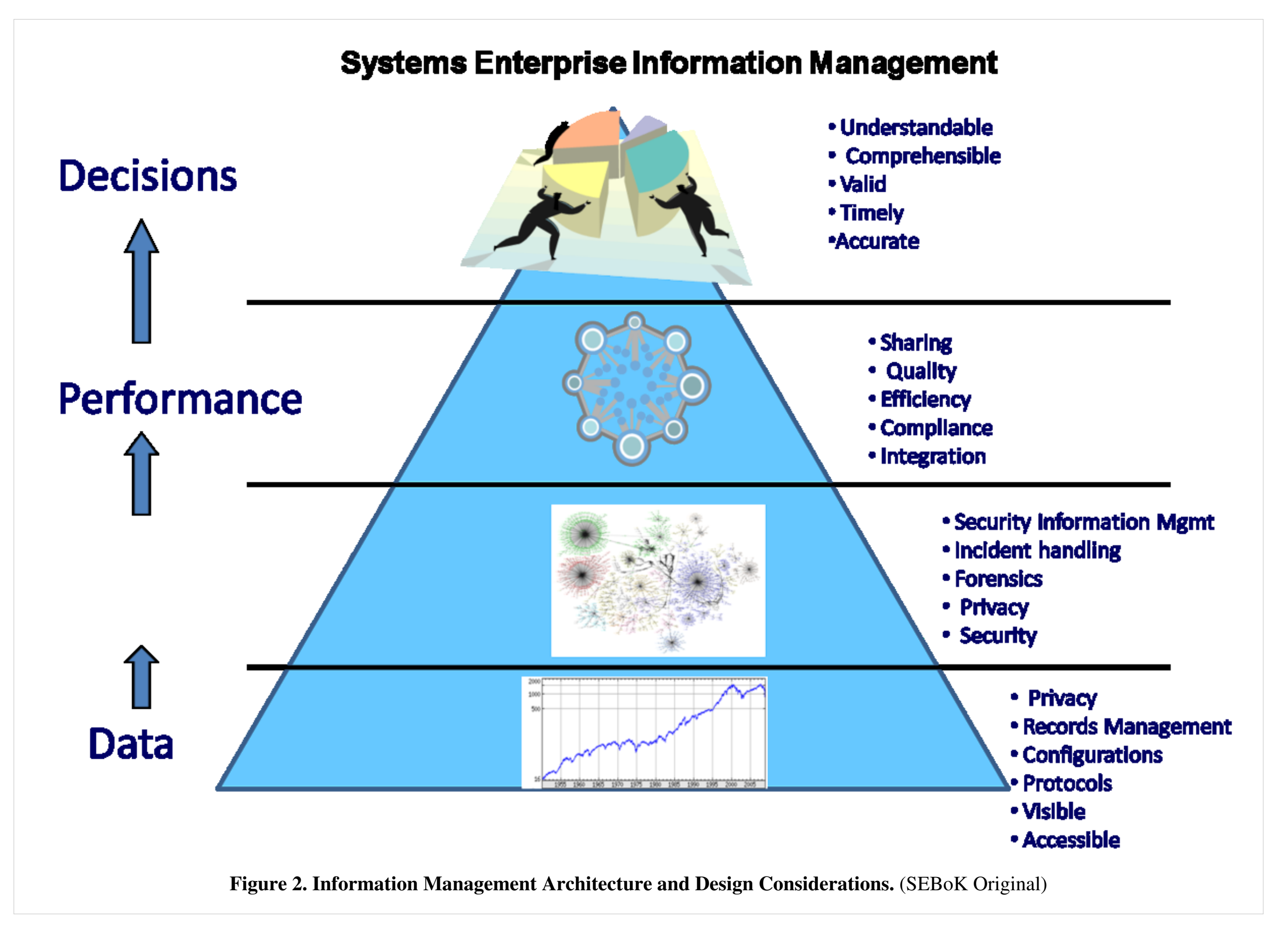


Figure 2. Information Management Architecture and Design Considerations. (SEBoK Original)

The effective and efficient employment of IM systems should solve business needs. These needs can center on several business objectives, such as efficiency, effectiveness, competitiveness, or profitability. From a business enterprise perspective, the systems engineer may be involved in several activities that support the development of IM systems, such as strategic planning, analyses of technology/business trends, development of applications, understanding operational disciplines, resource control techniques, and assessment of organization structures.

The IM process ensures that necessary information is created, stored, retained, protected, managed, and made easily available to those with a need and who are permitted access. It also ensures that information is disposed of when it is no longer relevant.

The Information Management Process

To quote from ISO/IEC/IEEE 15288 (2015):

> *The purpose of the Information Management Process is to generate, obtain, confirm, transform, retain, retrieve,* disseminate and dispose of information, to designated stakeholders..
>
> *Information management plans, executes, and controls the provision of information to designated stakeholders that is unambiguous, complete, verifiable, consistent, modifiable, traceable, and presentable.*

The first step in the IM process is to plan IM. The output of this step is the IM strategy or plan. The second step is to perform IM. The outputs of this step are the creation, population and maintenance of one or more information repositories, together with the creation and dissemination of information management reports.

Plan Information Management

Issues that should be considered when creating the IM strategy/plan include:

- Scope
  - What information has to be managed?
  - How long will the information need to be retained?
  - Is a system data dictionary is required to be able to "tag" information for ease of search and retrieval?
  - Will the media that will be used for the information to be managed be physical, electronic, or both?
  - Have a work in progress (WIP) and formally released information already been considered when establishing data repositories?
- Constraints
  - What level of configuration control that has to be applied to the information?
  - Are there any regulatory requirements relating to the management of information for this project; this could include export control requirements?
  - Are there any customer requirements or agreements relating to the management of project information?
  - Are there any industry standards relating to the management of project information?
  - Are there any organization/enterprise directives, procedures, or standards relating to the management of project information?
  - Are there any project directives, procedures, or standards relating to the management of project information?
- Control/Security
  - Who is allowed access to the information? This could include people working on the project, other members of the organization/enterprise, customers, partners, suppliers and regulatory authorities.
  - Are there requirements to protect the information from unauthorized access? This could include intellectual property (IP) rights that have to be respected - for instance if information from suppliers is to be stored and there is the possibility of a supplier gaining access to information belonging to a competitor who is also a supplier for the project.
  - What data repository or repositories are to be used?
    - Has the volume of information to be stored been considered when selecting repositories?
    - Has speed of access and search been considered when selecting repositories?
  - If electronic information is to be stored, what file formats are allowed?
  - Have requirements been defined to ensure that the information being stored is valid?
  - Have requirements been defined to ensure that information is disposed of correctly when it is no longer required to be stored, or when it is no longer valid? For instance, has a review period been defined for each piece of information?
- Life Cycle
  - If electronic information is to be stored for a long time, how will it be "future-proofed" – for instance, are neutral file formats available, or will copies of the software that created or used the information be retained?
  - Have disaster recovery requirements been considered – e.g., if a server holding electronic information is destroyed, are there back-up copies of the information? Are the back-up copies regularly accessed to show that information recovery is flawless?
  - Is there a formal requirement to archive designated information for compliance with legal (including regulatory), audit, and information retention requirements? If so, has an archive and archiving method been defined?
  - Some information may not be required to be stored (e.g., the results files for analyses when the information occupies a large volume and can be regenerated by the analysis tool and the input file). However, if the cost to re-generate the information is high, consider doing a cost/benefit analysis for storage versus regeneration.

Perform Information Management

Issues that should be considered when performing information management include:

- Is the information valid (is it traceable to the information management strategy/plan and the list of information to be managed)?
- Has the workflow for review and approval of information been defined to transfer information from "work in progress" to "released?"
- Are the correct configuration management requirements being applied to the information? Has the information been baselined?
- Have the correct "tags" been applied to the information to allow for easy search and retrieval?
- Have the correct access rules been applied to the information? Can users access the information that they are permitted to access, and only this information?
- If required, has the information been translated into a neutral file format prior to storage?
- Has a review date been set for assessing the continued validity of the information?
- Has the workflow for review and removal of unwanted, invalid, or unverifiable information (as defined in organization/enterprise policy, project policy, security or intellectual property requirements) been defined?
- Has the information been backed up and has the backup recovery system been tested?
- Has designated information been archived in compliance with legal (including regulatory), audit, and information retention requirements?
- Does the IM system satisfy defined performance requirements - for instance, speed of access, availability, and searchability?

Linkages to Other Systems Engineering Management Topics

The systems engineering IM process is closely coupled with the system definition, planning, and CM processes. The requirements for IM are elicited from stakeholders as part of the system definition process. What/when information is to be stored in the systems engineering lifecycle is defined in the planning process and configuration control requirements are defined in the CM process.

Practical Considerations

Key pitfalls and good practices related to IM are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in planning and performing IM are provided in Table 1:

Table 1. Information Management Pitfalls. (SEBoK Original)

Pitfall Name	Pitfall Description
No Data Dictionary	• Not defining a data dictionary for the project may result in inconsistencies in naming conventions for information and proliferation of meta-data "tags", which reduces the accuracy and completeness of searches for information and adding search time performance.
No Metadata	• Not "tagging" information with metadata or doing this inconsistently may result in searches being based on metadata tags, which are ineffective and can overlook key information.
No Back-Up Verification	• Not checking that information can be retrieved effectively from a back-up repository when access to the back-up is needed may result in one discovering that the back-up information is corrupted or not accessible.
Access Obsolescence	• This refers to saving information in an electronic format which eventually ceases to be accessible and not retaining a working copy of the obsolete software to be able to access the information.

Inadequate Long-Term Retention	• This refers to the error of archiving information on an electronic medium that does not have the required durability to be readable through the required retention life of the information and not regularly accessing and re-archiving the information.
Inadequate Validity Maintenance	• Not checking the continued validity of information results in outdated or incorrect information being retained and used.

Good Practices

Some good practices gathered from the references are provided in Table 2:

Table 2. Information Management Good Practices. (SEBoK Original)

Good Practice Name	Good Practice Description
Guidance	• The DAMA Guide to the Data Management Body of Knowledge provides an excellent, detailed overview of IM at both the project and enterprise level.
Information as an Asset	• Recognize that information is a strategic asset for the organization and needs to be managed and protected.
Information Storage Capacity	• Plan for the organization's information repository storage capacity to need to double every 12 to 18 months.
Effective Information Access	• Information that sits in a repository adds no value. It only adds value when it is used. So the right people need to be able to access the right information easily and quickly.
Data Modeling	• Invest time and effort in designing data models that are consistent with the underlying structure and information needs of the organization.
Quality Management	• The cost impact of using poor quality information can be enormous. Be rigorous about managing the quality of information.
Information Repository Design	• The impact of managing information poorly can also be enormous (e.g., violating intellectual property or export control rules). • Make sure that these requirements are captured and implemented in the information repository, and that all users of the repository are aware of the rules that they need to follow and the penalties for infringement.

References

Works Cited

ISO/IEC/IEEE. 2008. *Systems and Software Engineering - System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation/International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2008.

Mosley, M. (ed.). 2009. *The DAMA Guide to the Data Management Body of Knowledge (DAMA-DMBOK Guide)*. Bradley Beach, NJ, USA: Technics Publications.

Primary References

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

Mosley, M. (ed.). 2009. *The DAMA Guide to the Data Management Body of Knowledge (DAMA-DMBOK Guide).* Bradley Beach, NJ, USA: Technics Publications.

Redman, T. 2008. *Data Driven: Profiting from Your Most Important Business Asset*. Cambridge, MA, USA: Harvard Business Press.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Quality Management

Whether a systems engineer delivers a product, a service, or an enterprise, the deliverable should meet the needs of the customer and be fit for use. Such a deliverable is said to be of high quality. The process to assure high quality is called quality management.

Overview

Over the past 80 years, a *quality movement* has emerged to enable organizations to produce high quality deliverables. This movement has gone though four stages:

1. **Acceptance Sampling** was developed to apply statistical tests to assist in the decision of whether or not to accept a lot of material based on a random sample of its content.
2. **Statistical Process Control** (SPC) was developed to determine if production processes were stable. Instead of necessarily measuring products, processes are measured instead. Processes that departed from a state of statistical control were far more likely to develop low quality deliverables.
3. **Design for Quality** focused on designing processes that were robust against causes of variation, reducing the likelihood that a process would go out of control, and accordingly reducing the monitoring requirements.
4. **Six sigma** methods are applied the tools and power of statistical thinking to improve other aspects of the organization.

Definitions

The American Society for Quality [1] provides the following definitions:

- Acceptance Sampling involves the inspection of a sample to decide whether to accept the entire lot. There are two types of sampling:
  - In attributes sampling, the presence or absence of a characteristic is noted in each of the units inspected.
  - In variables sampling, the numerical magnitude of a characteristic is measured and recorded for each inspected unit. This involves reference to a continuous scale of some kind.
- SPC is the application of statistical techniques to control a process. It is often used interchangeably with the term “statistical quality control.”
- Quality is a subjective term for which each person or sector has its own definition. In technical usage, quality can have two meanings:
  - The characteristics of a product or service that bear on its ability to satisfy stated or implied needs.
  - A product or service free of deficiencies. According to Joseph Juran, quality means “fitness for use.” According to Philip Crosby, it means "conformance to requirements."

- Six Sigma is a method that provides organizations with tools to improve the capability of their business processes. This increase in performance and decrease in process variation leads to defect reduction and improvement in profits, employee morale, and quality of products or services. Six Sigma quality is a term generally used to indicate a process is well controlled (±6 s from the centerline in a control chart).

Quality Attributes

Quality attributes, also known as quality factors, quality characteristics, or non-functional requirements, are a set of system functional and non-functional requirements that are used to evaluate the system performance. There are a large number of system quality attributes identified in the literature (e.g. MSDN 2010, Barbacci et al. 1995). Depending on the type of the system being considered, some of these attributes are more prominent than others. Ideally, a system would be optimized for all the quality attributes that are important to the stakeholders, but this is an impossible task. Therefore, it is important to conduct a trade off analysis to identify the relationship between the attributes and establish whether a change in one attributes would positively or negatively affect any other attributes. An example of such trade off is shown in Table 1 below. (See SEBoK discussion on specialty engineering for additional information on quality attributes.)

Table 1. Attribute Trade Off. (SEBoK Original)

	Flexibility	Maintainability	Reliability
Flexibility		+	-
Maintainability	+		+
Reliability	-	+	

Finding the right set of quality attributes is the first step in quality control and management. In order to achieve high quality, quality has to be measured, monitored, managed, and improved on. Therefore, in order to increase the overall system quality, it is necessary to

- identify and prioritize the quality attributes
- identify the metrics that can be used for these attributes
- measure and monitor the attributes
- validate the measurements
- analyze the result of those measurements
- establish processes and procedures that result in improved system quality, based on the analysis.

Quality Attributes for Products

Quality attributes for a product focuses on the conformance to the specifications for the product; frequently these are manufacturing specifications. Examples include physical characteristics (length, weight, finish, capacity, etc.) being inside a given tolerance range. The physical characteristics can be related to the function of the product or to aesthetic qualities.

A single product may have a vector of quality attributes of high dimension as wells as an associated region in which the vector is expected to be. Often the quality is summarized by saying the item is "in compliance" (if the vector is in the acceptable region) or "defective" (if the vector is outside the acceptable region).

Quality Attributes for Services

Quality of services plays a major role in the customer satisfaction, which is the measurement of the overall system quality. Services can be divided into two major categories: primary and secondary. The city public transportation system, the U.S. postal service, or the medical services provided by a hospital are all examples of primary services. Services that provide help to a customer are secondary services, which are typically referred to as a *customer service*. Identifying the appropriate quality attributes is critical in the quality management of services. Some examples of service quality attributes include: affordability, availability, dependability, efficiency, predictability, reliability, responsiveness, safety, security, usability, etc. Again, depending on the type of the service, some of these attributes are more prominent than the others.

For example, in the case of services that are provided by the hospital, one may be more interested in the availability, reliability, and responsiveness than potentially the security (typically hospitals are assumed to be safe) and the affordability (typically insurance covers the majority of the cost). Of course, if the patient does not have a good insurance coverage, then the importance of affordability will increase (de Knoning, 2006).

Quality Attributes for Enterprises

An enterprise typically refers to a large complex set of interconnected entities that includes people, technologies, processes, financial, and physical elements. Clearly, a typical enterprise has a number of internal and external stakeholders, and as a result there are a large number of quality attributes that will define its quality. Identifying the right set of attributes is typically more challenging in such a complex system. An example of an enterprise is the air traffic management system that is mainly responsible for the safe and efficient operation of the civil aviation within a country or collection of countries. There are a large number of stakeholders that are concerned about the overall quality of the system, some example of these stakeholders and some of the primary quality attributes that they are concerned with are identified in Table 2.

Table 2. Enterprise Stakeholders and their Quality Attributes. (SEBoK Original)

Stakeholders	**Primary Quality Attributes**
Passengers	Ssfety, affordability, and reliability
Airlines	adaptability, efficiency, and profitability
Air Traffic Controller	safety, reliability, and usability
Hardware & Software Developers	reliability, fault tolerance, and maintainability
Government/Regulatory Agency	safety, reliability, affordability, etc.

Measuring Quality Attributes

Quality cannot be achieved if it cannot be measured. The Measurement System Analysis (MSA) (Wheeler and Lynday 1989) is a set of measuring instruments that provide an adequate capability for a team to conduct appropriate measurements in order to monitor and control quality. The MSA is a collection of

- **Tools** - measuring instruments, calibration, etc.
- **Processes** - testing and measuring methods, set of specifications, etc.
- **Procedures** - policies and procedures and methodologies that are defined by the company and/or regulatory agency
- **People** - personnel (managers, testers, analysis, etc.) who are involved in the measurement activities
- **Environment** - both environmental setting and physical setting that best simulate the operational environment and/or the best setting to get the most accurate measurements

Once the quality attributes are identified and prioritized, then the MSA supports the monitor and control of overall system quality.

Additional details about measurement are presented in the measurement article.

Quality Management Strategies

Acceptance Sampling

In acceptance sampling many examples of a product are presented for delivery. The consumer samples from the lot and each member of the sample is then categorized as either *acceptable* or *unacceptable* based on an attribute (attribute sampling) or measured against one or more metrics (variable sampling). Based on the measurements, an inference is made as to whether the lot meets the customer requirements.

There are four possible outcomes of the sampling of a lot, as shown in Table 3.

Table 3. Truth Table - Outcomes of Acceptance Sampling. (SEBoK Original)

	Lot Meets Requirement	Lot Fails Requirement
Sample Passes Test	No error	Consumer risk
Sample Fails Test	Producer risk	No error

A sample acceptance plan balances the risk of error between the producer and consumer. Detailed ANSI/ISO/ASQ standards describe how this allocation is performed (ANSI/ISO/ASQ A3534-2-1993: *Statistics—Vocabulary and Symbols—Statistical Quality Control*).

Statistical Process Control

SPC is a method that was invented by Walter A. Shewhart (1931) that adopts statistical thinking to monitor and control the behaviors and performances of a process. It involves using statistical analysis techniques as tools in appropriate ways, such as providing an estimate of the variation in the performance of a process, investigating the causes of this variation, and offering the engineer the means to recognize from the data when the process is not performing as it should (Mary et al. 2006, 441). In this context, *performance* is measured by how well the process is performed.

The theory of quality management emphasizes managing processes by fact and maintaining systematic improvement. All product developments are a series of interconnected processes that have variation in their results. Understanding variation with SPC technology can help the process executors understand the facts of their processes and find the improvement opportunities from a systematic view.

Control charts are common tools in SPC. The control chart is also called the Shewhart 3-sigma chart. It consists of 3 limit lines: the center line, which is the mean of statistical samples, and the upper and lower control limit lines, which are calculated using the mean and standard deviation of statistical samples. The observed data points or their statistical values are drawn in the chart with time or other sequence orders. Upper and lower control limits indicate the thresholds at which the process output will be considered as *unlikely*. There are two sources of process variation. One is common cause variation, which is due to inherent interaction among process components. Another is assignable cause, which is due to events that are not part of the normal process. SPC stresses bringing a process into a state of statistical control, where only common cause variation exists, and keeping it in control. A control chart is used to distinguish between variation in a process resulting from common causes and assignable causes.

If the process is in control, and if standard assumptions are met, points will demonstrate a normal distribution around the control limit. Any points outside the either of the limits, or in systematic patterns imply a new source of variation would be introduced. A new variation means increased quality cost. Additional types of control charts exist,

including: cumulative sum charts that detect small, persistent step change model departures and moving average charts, which use different possible weighting schemes to detect persistent changes (Hawkins and Olwell 1996).

Design for Quality

Variation in the inputs to a process usually results in variation in the outputs. Processes can be designed, however, to be robust against variation in the inputs. Response surface experimental design and analysis is the statistical technique that is used to assist in determining the sensitivity of the process to variations in the input. Such an approach was pioneered by Taguchi.

Six Sigma

Six sigma methodology (Pyzdek and Keller, 2009) is a set of tools to improve the quality of business processes; in particular, to improve performance and reduce variation. Six sigma methods were pioneered by Motorola and came into wide acceptance after they were championed by General Electric.

Problems resulting in variation are addressed by six sigma projects, which follow a five-stage process:

1. **Define** the problem, the stakeholders, and the goals.
2. **Measure** key aspects and collect relevant data.
3. **Analyze** the data to determine cause-effect relationships.
4. **Improve** the current process or **design** a new process.
5. **Control** the future state or **verify** the design.

These steps are known as **DMAIC** for existing processes and **DMADV** for new processes. A variant of six sigma is called lean six sigma wherein the emphasis is on improving or maintaining quality while driving out waste.

Standards

Primary standards for quality management are maintained by ISO, principally the IS0 9000 series [2]. The ISO standards provide requirements for the quality management systems of a wide range of enterprises, without specifying how the standards are to be met and also have world-wide acceptance. The key requirement is that the system must be audited.

In the United States, the Malcolm Baldridge National Quality Award presents up to three awards in six categories: manufacturing, service company, small business, education, health care, and nonprofit. The Baldridge Criteria [3] have become de facto standards for assessing the quality performance of organizations.

References

Works Cited

Barbacci, M., M.H. Klein, T.A. Longstaff, and C.B. Weinstock. 1995. *Quality Attributes.* Pittsburg, PA, USA: Software Engineering Institute/Carnegie Melon University. CMU/SEI-95-TR-021.

Chrissis, M.B., M. Konrad, and S. Shrum. 2006. *CMMI for Development: Guidelines for Process Integration and Product Improvement,* 2nd ed. Boston, MA, USA: Addison Wesley.

Evans, J. and W. Lindsay. 2010. *Managing for Quality and Performance Excellence.* Florence, KY, USA: Cengage Southwestern.

Juran, J.M. 1992. *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services.* New York, NY, USA: The Free Press.

Koning, H. de, J.P.S. Verver, J. van den Heuvel, S. Bisgaard, R.J.M.M. Does. 2006. "Lean Six Sigma in Healthcare." *Journal for Healthcare Quality.* 28(2) pp 4-11.

MSDN. 2010. "Chapter 16: Quality Attributes," in *Microsoft Application Architecture Guide*, 2nd Edition. Microsoft Software Developer Network, Microsoft Corporation. Accessed August 31, 2012. Available online at http://msdn.microsoft.com/en-us/library/ff650706.

Moen, R.D., T.W. Nolan, and L.P. Provost. 1991. *Quality Improvement through Planned Experimentation.* New York, NY, USA: McGraw-Hill.

Pyzdek, T. and P.A. Keller. 2009. *The Six Sigma Handbook,* 3rd ed. New York, NY: McGraw-Hill.

Shewhart, W.A. 1931. *Economic Control of Manufactured Product*. New York, NY, USA: Van Nostrand.

Wheeler, D.J. and R.W. Lyday. 1989. *Evaluating the Measurement Process,* 2nd ed. Knoxville, TN, USA: SPC Press.

Primary References

Chrissis, M.B, M. Konrad, S. Shrum. 2011. *CMMI for Development: Guidelines for Process Integration and Product Improvement*, 3rd ed. Boston, MA, USA: Addison-Wesley Professional.

Evans, J. and W. Lindsay. 2010. *Managing for Quality and Performance Excellence.* Florence, KY, USA: Cengage Southwestern.

Juran, J.M. 1992. *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services.* New York, NY, USA: The Free Press.

Moen, R.D., T.W. Nolan, and L.P. Provost. 1991. *Quality Improvement through Planned Experimentation.* New York, NY, USA: McGraw-Hill.

Pyzdek, T. and P.A. Keller. 2009. *The Six Sigma Handbook,* 3rd ed. New York, NY: McGraw-Hill.

Wheeler, D.J. and R.W. Lyday. 1989. *Evaluating the Measurement Process,* 2nd ed. Knoxville, TN, USA: SPC Press.

Additional References

Hawkins, D. and D.H. Olwell. 1996. *Cumulative Sum Charts and Charting for Quality Improvement*. New York, NY, USA: Springer.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://asq.org/glossary/index.html
[2] http://www.iso.org/iso/iso_catalogue/management_and_leadership_standards/quality_management.htm
[3] http://www.nist.gov/baldrige/publications/criteria.cfm

Knowledge Area: Product and Service Life Management

Product and Service Life Management

Product and service life management deals with the overall life cycle planning and support of a system. The life of a product or service spans a considerably longer period of time than the time required to design and develop the system. Systems engineers need to understand and apply the principles of life management throughout the life cycle of the system. (See Life Cycle Models for a general discussion of life cycles.) Specifically, this knowledge area (KA) focuses on changes to a system after deployment, including extension, modernization, disposal, and retirement.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Service Life Extension
- Capability Updates, Upgrades, and Modernization
- Disposal and Retirement

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

Overview

Product and service life management is also referred to as *system sustainment*. Sustainment involves the supportability of operational systems from the initial procurement to disposal. Sustainment is a key task for systems engineering that influences product and service performance and support costs for the entire life of the program.

Sustainment activities include: design for maintainability, application of built-in test, diagnostics, prognostics and other condition-based maintenance techniques, implementation of logistics footprint reduction strategies, identification of technology insertion opportunities, identification of operations and support cost reduction opportunities, and monitoring of key support metrics. Life cycle sustainment plans should be created for large, complex systems (DAU 2010). Product and service life management applies to both commercial systems (e.g. energy generation and distribution systems, information management systems, the Internet, and health industries) and government systems (e.g. defense systems, transportation systems, water-handling systems, and government services).

It is critical that the planning for system life management occur during the requirements phase of system development. (See System Requirements and System Definition). The requirements phase includes the analysis of life cycle cost alternatives, as well as gaining the understanding of how the system will be sustained and modified once it is operational.

The body of knowledge associated with product and service life management includes the following areas:

1. Service Life Extension - Systems engineers need to understand the principles of service life extension, the challenges that occur during system modifications, and issues involved with the disposal and retirement after a system has reached the end of its useful life.

2. Modernization and Upgrades - Managing service life extension uses the engineering change management process with an understanding of the design life constraints of the system. Modernizing existing legacy systems requires special attention and understanding of the legacy requirements and the importance of having a complete inventory of all the system interfaces and technical drawings.
3. Disposal and Retirement - Disposal and retirement of a product after reaching its useful life requires attention to environmental concerns, special handling of hazardous waste, and concurrent operation of a replacement system as the existing system is being retired.

Principles and Standards

The principles of product and service life management apply to different types of systems and domains. The type of system (commercial or government) should be used to select the correct body of knowledge and best practices that exist in different domains. For example, U.S. military systems would rely on sustainment references and best practices from the Department of Defense (DoD) (e.g., military services, Defense Acquisition University (DUA), etc.) and military standardization bodies (e.g., the American Institute of Aeronautics and Astronautics (AIAA), the Society of Automotive Engineers (SAE), the Society of Logistics Engineers (SOLE), the Open Geospatial Consortium (OGC), etc.).

Commercial aviation, power distribution, transportation, water-handling systems, the Internet, and health industries would rely on system life management references and best practices from a combination of government agencies, local municipalities, and commercial standardization bodies and associations (e.g., in the U.S.- the Department of Transportation (DOT), State of Michigan, International Organization for Standardization (ISO), Institute of Electrical and Electronics Engineers (IEEE), International Council on Systems Engineering (INCOSE), etc.).

Some standardization bodies have developed system life management practices that bridge both military and commercial systems (e.g., INCOSE, SOLE, ISO, IEEE, etc.). There are multiple commercial associations involved with defining engineering policies, best practices, and requirements for commercial product and service life management. Each commercial association has a specific focus for the market or domain area where the product is used. Examples of such commercial associations in the U.S. include: American Society of Hospital Engineering (ASHE); Association of Computing Machinery (ACM); American Society of Mechanical Engineers (ASME); American Society for Testing & Materials (ASTM) International; National Association of Home Builders (NAHB); and Internet Society (ISOC), including Internet Engineering Task Force (IETF), and SAE.

In addition, there are several specific resources which provide useful information on product and service life management:

- The *INCOSE Systems Engineering Handbook*, version 3.2.2, identifies several relevant points regarding product and service life management (2011).
- The *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)*, version 1.1, provides guidance on product changes and system retirement (Caltrans and USDOT 2005).
- *Systems Engineering and Analysis* emphasizes design for supportability and provides a framework for product and service supportability and planning for system retirement (Blanchard and Fabrycky 2006).
- *Modernizing Legacy Systems* identifies strategies for product and service modernization (Seacord, Plakosh, and Lewis 2003).
- "Logistics and Materiel Readiness" (http://www.acq.osd.mil/log/ [1]) provides online policies, procedures, and planning references for product service life extension, modernization, and retirement (OUSD(AT&L) 2011).
- *A Multidisciplinary Framework for Resilience to Disasters and Disruptions* provides insight into architecting a system for extended service life (Jackson 2007).

Good Practices

Major pitfalls associated with systems engineering (SE) after the deployment of products and services can be avoided if the systems engineer:

- Recognizes that the systems engineering process does not stop when the product or service becomes operational.
- Understands that certain life management functions and organizations, especially in the post-delivery phase of the life cycle, are part of the systems engineering process.
- Identifies that modifications need to comply with the system requirements.
- Considers that the users must be able to continue the maintenance activities drawn up during the system requirement phase after an upgrade or modification to the system is made.
- Accounts for changing user requirements over the system life cycle.
- Adapts the support concepts, drawn up during development, throughout the life cycle.
- Applies engineering change management to the total system.

Not addressing these areas of concern early in development and throughout the product or service's life cycle can have dire consequences.

References

Works Cited

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. "Acquisition Community Connection (ACC): Where the DoD AT&L workforce meets to share knowledge." Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense (DoD). https://acc.dau.mil.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. "A Multidisciplinary Framework for Resilience to Disasters and Disruptions." *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. "Logistics and Materiel Readiness On-line policies, procedures, and planning references." Arlington, VA, USA: Office of the Under Secretary of Defense for Aquisition, Transportation and Logistics (OUSD(AT&L). http://www.acq.osd.mil/log/http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson, S. 2007. "A Multidisciplinary Framework for Resilience to Disasters and Disruptions." *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. "Logistics and Materiel Readiness On-line policies, procedures, and planning references." Arlington, VA, USA: Office of the Under Secretary of Defense for Aquisition, Transportation and Logistics (OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

Additional References

Blanchard, B.S. 2010. *Logistics engineering and management,* 5th ed. Englewood Cliffs, NJ, USA: Prentice Hall: 341-342.

Braunstein, A. 2007. "Balancing Hardware End-of-Life Costs and Responsibilities." Westport, CT, USA: Experture Group, ETS 07-12-18.

Brown, M., R. Weyers, and M. Sprinkel. 2006. "Service Life Extension of Virginia Bridge Decks afforded by Epoxy-Coated Reinforcement." *Journal of ASTM International (JAI).* 3(2): 13.

DLA. 2010. "Defense logistics agency disposition services." In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla.mil.

EPA. 2010. "Wastes In U.S. Environmental Protection Agency (EPA)." Washington, D.C. Available at: http://www.epa.gov/epawaste/index.htm.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

FSA. 2010. "Template for 'System Retirement Plan' and 'System Disposal Plan'." In Federal Student Aid (FSA)/U.S. Department of Eduation (DoEd). Washington, DC, USA. Accessed August 5, 2010. Available at: http://federalstudentaid.ed.gov/business/lcm.html.

Gehring, G., D. Lindemuth, and W.T. Young. 2004. "Break Reduction/Life extension Program for CAST and Ductile Iron Water Mains." Paper presented at NO-DIG 2004, Conference of the North American Society for Trenchless Technology (NASTT), March 22-24, New Orleans, LA, USA.

Hovinga, M.N., and G.J. Nakoneczny. 2000. "Standard Recommendations for Pressure Part Inspection during a Boiler Life Extension Program." Paper presented at ICOLM (International Conference on Life Management and Life Extension of Power Plant), May, Xi'an, P.R. China.

IEC. 2007. *Obsolescence Management - Application Guide*, ed 1.0. Geneva, Switzerland: International Electrotechnical Commission, IEC 62302.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical

and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Ihii, K., C.F. Eubanks, and P. Di Marco. 1994. “Design for Product Retirement and Material Life-Cycle.” *Materials & Design.* 15(4): 225-33.

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook,* version 3.2, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. *Systems Engineering Body of Knowledge, provisional,* version 2.0. Singapore.

Jackson, S. 1997. *Systems Engineering for Commercial Aircraft.* Surrey, UK: Ashgate Publishing, Ltd.

Koopman, P. 1999. “Life Cycle Considerations.” Pittsburgh, PA, USA:Carnegie Mellon. Accessed August 5, 2010. Available at: http://www.ece.cmu.edu/~koopman/des_s99/life_cycle/index.html.

L3 Communications. 2010. “Service Life Extension Program (SLEP).” Newport News, VA, USA: L3 Communications, Flight International Aviation LLC.

Livingston, H. 2010. “GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices.” McClellan, CA, USA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Minneapolis-St. Paul Chapter of SOLE. 2003. "Systems Engineering in Systems Deployment and Retirement, presented to INCOSE." Minneapolis-St. Paul, MN, USA: International Society of Logistics (SOLE), Minneapolis-St. Paul Chapter.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C.: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Nguyen, L. 2006. “Adapting the Vee Model to Accomplish Systems Engineering on Change Projects.” Paper presented at 9th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference, San Diego, CA, USA.

Office of Natural Gas and Oil Technology. 1999. *Reservoir LIFE Extension Program: Encouraging Production of Remaining Oil and Gas.* Washington, DC, USA: U.S. Department of Engery (DoE).

Paks Nuclear Power Plant. 2010. “Paks Nuclear Power Plant: Service Life Extension.” In Paks Nuclear Power Plant, Ltd.. Hungary, accessed August 5, 2010. Available at: http://paksnuclearpowerplant.com/service-life-extension.

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transpofration (NDDOT).

SAE International. 2010. “Standards: Commercial Vehicle--Maintenance and Aftermarket.” Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

SAE International. 2010. “Standards: Maintenance and Aftermarket.” Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

Sukamto, S. 2003. “Plant Aging and Life Extension Program at Arun LNG Plant Lhokseumawe, North Aceh, Indonesia.” Paper presented at 22nd Annual World Gas Conference, June 1-5, Tokyo, Japan.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.acq.osd.mil/log/

Service Life Extension

Product and service life extension involves continued use of a product and/or service beyond its original design life. Product and service life extension involves assessing the risks and the life cycle cost (LCC) of continuing the use of the product or service versus the cost of a replacement system.

Service life extension (SLE) emphasizes reliability upgrades and component replacement or rebuilding of the system to delay the system's entry into *wear-out* status due to issues such as expensive sustainment, reliability, safety, and/or performance requirements that can no longer be met. The goal is typically to return the system to as new a condition as possible while remaining consistent with the economic constraints of the program.

SLE is regarded as an environmentally friendly way to relieve rampant waste by prolonging the useful life of retiring products and preventing them from being discarded too early when they still have unused value. However, challenged by fast-changing technology and physical deterioration, a major concern in planning a product SLE is considering to what degree a product or service is fit to have its life extended.

Topic Overview

SLE is typically required in the following circumstances:

- The system no longer meets the system performance or reliability requirements.
- The cost of operation and maintenance exceeds the cost of SLE, or the available budgets.
- Parts are no longer available for repair and maintenance.
- Operation of the system violates rules or regulations, such as environmental or safety regulations.
- Parts of the system are about to reach their operations life limits, which will result in the issue listed above occurring.

It is best if systems engineers use a pro-active approach that predicts ahead, so that SLE can be accomplished before the system fails to meet its requirements and before the operations and support costs rise above acceptable limits.

Key factors that must be considered by the systems engineer during service life extension include

- current life cycle costs of the system
- design life and expected remaining useful life of the system
- software maintenance
- configuration management
- warranty policy
- availability of parts, subsystems, and manufacturing sources
- availability of system documentation to support life extension

System design life is a major consideration for SLE. System design life parameters are established early on during the system design phase and include key assumptions involving safety limits and material life. Safety limits and the properties of material aging are critical to defining system life extension. Jackson emphasizes the importance of architecting for system resiliency in increases system life. He also points out that a system can be architected to withstand internal and external disruptions (2007, 91-108). Systems that age through use, such as aircraft, bridges,

and nuclear power plants, require periodic inspection to ascertain the degree of aging and fatigue. The results of inspections determine the need for actions to extend the product life (Elliot, Chen, and Swanekamp 1998, sec. 6.5).

Software maintenance is a critical aspect of SLE. The legacy system may include multiple computer resources that have been in operation for a period of many years and have functions that are essential and must not be disrupted during the upgrade or integration process. Typically, legacy systems include a computer resource or application software program that continues to be used because the cost of replacing or redesigning it is prohibitive. The Software Engineering Institute (SEI) has addressed the need for SLE of software products and services and provides useful guidance in the on-line library for Software Product Lines (SEI 2010, 1). (See Systems Engineering and Software Engineering for additional discussion of software engineering (SwE) factors to consider.)

Systems engineers have found that service life can be extended through the proper selection of materials. For example, transportation system elements such as highway bridges and rail systems are being designed for extended service life by using special epoxy-coated steel (Brown, Weyers, and Spinkel 2006, 13). Diminishing manufacturing sources and diminishing suppliers need to be addressed early on in the SLE process. Livingston (2010) in *Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices* provides a method for addressing product life extension when the sources of supply are an issue. He addresses the product life cycle model and describes a variety of methods that can be applied during system design to minimize the impact of future component obsolescence issues.

During product and service life extension, it is often necessary to revisit and challenge the assumptions behind any previous life cycle cost analysis (and constituent analyses) to evaluate their continued validity and/or applicability early in the process.

Application to Product Systems

Product life extension requires an analysis of the LCC associated with continued use of the existing product versus the cost of a replacement product. In the INCOSE Systems Engineering Handbook, Chapter 3.3 points out that the support stage includes service life extension (2012). Chapter 7 provides a framework to determine if a product’s life should be extended (INCOSE 2012). In Systems Engineering and Analysis, Chapter 17 provides a LCC methodology and emphasizes the analysis of different alternatives before making a decision on product life extension (Blanchard and Fabrycky 2011).

For military systems, service life extension is considered a subset of modification or modernization. Military systems use well-developed and detailed guidance for SLE programs (SLEP). The Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics (OSD AT&L) provides an online reference for policies, procedures, planning guidance, and whitepapers for military product service life extension (DAU 2011). Continuous military system modernization is a process by which state-of-the-art technologies are inserted continuously into weapon systems to increase reliability, lower sustainment costs, and increase the war fighting capability of a military system to meet evolving customer requirements throughout an indefinite service life.

Aircraft service life can be extended by reducing the dynamic loads which lead to structural fatigue. The Boeing B-52 military aircraft and the Boeing 737 commercial aircraft are prime examples of system life extension. The B-52 was first fielded in 1955 and the Boeing 737 has been fielded since 1967; both aircraft are still in use today.

For nuclear reactors, system safety is the most important precondition for service life extension. System safety must be maintained while extending the service life (Paks 2010). Built-in tests, automated fault reporting and prognostics, analysis of failure modes, and the detection of early signs of wear and aging may be applied to predict the time when maintenance actions will be required to extend the service life of the product. (For additional discussion, see Safety Engineering.)

Application to Service Systems

For systems that provide services to a larger consumer base, SLE involves continued delivery of the service without disrupting consumer use. This involves capital investment and financial planning, as well as a phased deployment of changes. Examples of these concepts can be seen in transportation systems, water treatment facilities, energy generation and delivery systems, and the health care industry. As new technologies are introduced, service delivery can be improved while reducing LCC's. Service systems have to continuously assess delivery costs based upon the use of newer technologies.

Water handling systems provide a good example of a service system that undergoes life extension. Water handling systems have been in existence since early civilization. Since water handling systems are in use as long as a site is occupied (e.g., the Roman aqueducts) and upgrades are required as the population expands, such systems are a good example of "systems that live forever." For example, there are still U.S. water systems that use a few wooden pipes since there has been no reason to replace them. Water system life extension must deal with the issue of water quality and the capacity for future users (Mays 2000). Water quality requirements can be further understood from the AWWA Manuals of Water Supply Practices (AWWA 2010).

Application to Enterprises

SLE of a large enterprise, such as the National Astronautics and Space Administration's (NASA) national space transportation system, involves SLE on the elements of the enterprise, such as the space vehicle (shuttle), ground processing systems for launch operations and mission control, and space-based communication systems that support space vehicle tracking and status monitoring. SLE of an enterprise requires a holistic look across the entire enterprise. A balanced approach is required to address the cost of operating older system components versus the cost required to implement service life improvements.

Large enterprise systems, such as oil and natural gas reservoirs, which span broad geographical areas, can use advanced technology to increase their service life. The economic extraction of oil and natural gas resources from previously established reservoirs can extend their system life. One such life extension method is to pump special liquids or gases into the reservoir to push the remaining oil or natural gas to the surface for extraction (Office of Natural Gas & Oil Technology 1999).

Other Topics

Commercial product developers have been required to retain information for extended periods of time after the last operational product or unit leaves active service (for up to twenty years). Regulatory requirements should be considered when extending service life (INCOSE 2012).

Practical Considerations

The cost associated with life extension is one of the main inputs in the decision to extend service life of a product or a service. The cost of SLE must be compared to the cost of developing and deploying a new system, as well as the functional utility the user will obtain from each of the alternatives. It is often the case that the funding required for SLE of large complex systems is spread over several fiscal planning cycles and is therefore subject to changes in attitude by the elected officials that appropriate the funding.

The challenges with upgrading a system while it is still being used, which is often the case with SLE, must be understood and planned to avoid serious disruptions to the services the systems provide.

Any SLE must also consider the obsolescence of the systems parts, (e.g., software, amount of system redesign that is required to eliminate the obsolete parts, etc.).

References

Works Cited

AWWA. 2010. “AWWA Manuals of Water Supply Practices.” In American Water Works Association (AWWA). Denver, CO.Accessed August 5, 2010. Available at: http:/ / www. awwa. org/ Resources/ standards. cfm?ItemNumber=47829&navItemNumber=47834.

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Brown, M., R. Weyers, and M. Sprinkel. 2006. “Service Life Extension of Virginia Bridge Decks afforded by Epoxy-Coated Reinforcement.” *Journal of ASTM International (JAI),* 3(2): 13.

DAU. 2010. “Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge.” In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Integrated Design and Process Science.* 11(2).

Livingston, H. 2010. “GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices.” McClellan, CA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

Office of Natural Gas and Oil Technology. 1999. *Reservoir LIFE Extension Program: Encouraging Production of Remaining Oil and Gas.* Washington, DC, USA: U.S. Department of Engery (DoE).

Paks Nuclear Power Plant. 2010. “Paks Nuclear Power Plant: Service Life Extension.” In Paks Nuclear Power Plant, Ltd.. Hungary, accessed August 5, 2010. Available at: http://paksnuclearpowerplant.com/service-life-extension.

SEI. 2010. “Software Engineering Institute.” In Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU). Pittsburgh, PA, accessed August 5, 2010. http://www.sei.cmu.edu.

Primary References

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. “Logistics and Materiel Readiness On-line policies, procedures, and planning references.” Arlington, VA, USA: Office of the Under Secretary of Defense for Aquisition, Transportation and Logistics

(OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

Additional References

AWWA. 2010. “AWWA Manuals of Water Supply Practices.” In American Water Works Association (AWWA). Denver, CO. Accessed August 5, 2010. Available at: http:/ / www. awwa. org/ Resources/ standards. cfm?ItemNumber=47829&navItemNumber=47834.

Blanchard, B.S. 2010. *Logistics engineering and management*, 5th ed. Englewood Cliffs, NJ, USA: Prentice Hall, 341-342.

Braunstein, A. 2007. “Balancing Hardware End-of-Life Costs and Responsibilities.” Westport, CT, USA: Experture Group, ETS 07-12-18.

Brown, M., R. Weyers, and M. Sprinkel. 2006. “Service Life Extension of Virginia Bridge Decks afforded by Epoxy-Coated Reinforcement.” *Journal of ASTM International (JAI),* 3(2): 13.

Caltrans and USDOT. 2005. *Systems engineering guidebook for ITS,* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1: 278, 101-103, 107.

Casetta, E. 2001. *Transportation Systems Engineering: Theory and methods*. New York, NY, USA: Kluwer Publishers Academic, Springer.

DAU. 2010. “Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge.” In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

DLA. 2010. “Defense logistics agency disposition services.” In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla. mil.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

FAA. 2006. "Section 4.1" in “Systems Engineering Manual.” Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. “Radio and Television Broadcast Rules.” Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

Gehring, G., D. Lindemuth, and W.T. Young. 2004. “Break Reduction/Life extension Program for CAST and Ductile Iron Water Mains.” Paper presented at NO-DIG 2004, Conference of the North American Society for Trenchless Technology (NASTT), March 22-24, New Orleans, LA, USA.

Hovinga, M.N. and G.J. Nakoneczny. 2000. “Standard Recommendations for Pressure Part Inspection during a Boiler Life Extension Program.” Paper presented at ICOLM (International Conference on Life Management and Life Extension of Power Plant), May, Xi’an, P.R. China.

IEC. 2007. *Obsolescence Management - Application Guide*, ed 1.0. Geneva, Switzerland: International Electrotechnical Commission, IEC 62302.

IEEE. 2010. *IEEE Standard Framework for Reliability Prediction of Hardware*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1413.

IEEE. 1998. *IEEE Standard Reliability Program for the Development and Production of Electronic Systems and Equipment*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1332.

IEEE. 2008. *IEEE Recommended practice on Software Reliability*. New York: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1633.

IEEE 2005. *IEEE Standard for Software Configuration Management Plans*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 828.

IEEE. 2010. IEEE Standard Framework for Reliability Prediction of Hardware. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1413.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Ihii, K., C.F. Eubanks, and P. Di Marco. 1994. "Design for Product Retirement and Material Life-Cycle." *Materials & Design.* 15(4): 225-33.

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook,* version 3.2, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. "Systems Engineering Body of Knowledge, provisional," version 2.0. Singapore.

ISO/IEC. 2003. "Industrial Automation Systems Integration-Integration of Life-Cycle Data for Process Plants including Oil, Gas Production Factilies." Geneva, Switzerland: International Organization for Standardization (ISO)/International Electro technical Commission (IEC).

ISO/IEC. 1997. "Systems Engineering for Commercial Aircraft." Surrey, UK: Ashgate Publishing Ltd.

Koopman, P. 1999. "Life Cycle Considerations." In Carnegie-Mellon University (CMU). Pittsburgh, PA, USA, accessed August 5, 2010. Available at: http://www.ece.cmu.edu/~koopman/des_s99/life_cycle/index.html.

L3 Communications. 2010. "Service Life Extension Program (SLEP)." Newport News, VA, USA: L3 Communications, Flight International Aviation LLC.

Livingston, H. 2010. "GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices." McClellan, CA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methdology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C.: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Office of Natural Gas and Oil Technology. 1999. *Reservoir LIFE Extension Program: Encouraging Production of Remaining Oil and Gas.* Washington, DC, USA: U.S. Department of Engery (DoE).

Paks Nuclear Power Plant. 2010. "Paks Nuclear Power Plant: Service Life Extension." In Paks Nuclear Power Plant, Ltd.. Hungary, accessed August 5, 2010. Available at: http://paksnuclearpowerplant.com/service-life-extension.

Reason, J. 1997. *Managing the Risks of Organizational Accident.* Aldershot, UK: Ashgate.

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transpofration (NDDOT).

SAE International. 2010. “Standards: Automotive--Maintenance and Aftermarket.” Warrendale, PA: Society of Automotive Engineers (SAE) International.

Schafer, D.L. 2003. “Keeping Pace With Technology Advances When Funding Resources Are Diminished.” Paper presented at Auto Test Con. IEEE Systems Readiness Technology Conference, Anaheim, CA, USA: 584.

SEI. 2010. “Software Engineering Institute.” In Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU). Pittsburgh, PA, accessed August 5, 2010. http://www.sei.cmu.edu.

SOLE. 2009. “Applications Divisons.” In The International Society of Logistics (SOLE). Hyattsville, MD, USA, accessed August 5, 2010. http://www.sole.org/appdiv.asp.

Sukamto, S. 2003. “Plant Aging and Life Extension Program at Arun LNG Plant Lhokseumawe, North Aceh, Indonesia.” Paper presented at 22nd Annual World Gas Conference, June 1-5, Tokyo, Japan.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Capability Updates, Upgrades, and Modernization

Modernization and upgrades involve changing the product or service to include new functions and interfaces, improve system performance, and/or improve system supportability. The logistic support of a product or service reaches a point in its life where system modernization is required to resolve supportability problems and to reduce operational costs. The *INCOSE Systems Engineering Handbook* (INCOSE 2012) and *Systems Engineering and Analysis* (Blanchard and Fabrycky 2005) both stress the importance of using life cycle costs (LCC) when determining if a product or service should be modernized. Systems can be modernized in the field or returned to a depot or factory for modification.

Design for system modernization and upgrade is an important part of the system engineering process and should be considered as part of the early requirements and design activities. Engineering change proposals (ECPs) are used to initiate updates and modifications to the original system. Product and service upgrades can include new technology insertion, removing old equipment, or adding new equipment. Form, fit, function, and interface (F3I) is an important principle for upgrades where backward compatibility is a requirement.

Topic Overview

Product and service modernization involves the same systems engineering (SE) processes and principles that are employed during the upfront design, development, integration, and testing. The primary differences between product and service modernization are the various constraints imposed by the existing system architecture, design, and components. Modernizing a legacy system requires a detailed understanding of the product or service prior to making any changes. The constraints and the existence of design and test artifacts make it necessary for the systems engineers performing modernization to tailor the traditional development processes to fit the situation.

Product and service modernization occurs for many reasons, including the following:

1. The system or one of its subsystems is experiencing reduced performance, safety, or reliability.
2. A customer or other stakeholder desires a new capability for the system.
3. Some system components may be experiencing obsolescence, including the lack of spare parts.
4. New uses for the system require modification to add capabilities not built into the originally deployed system.

The first three reasons above are discussed in more detail in *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook.* (INCOSE UK Chapter 2010).

The UK chapter of the INCOSE developed *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook.* (INCOSE UK Chapter 2010) This guidance document applies to any system for which multiple systems are produced. These systems may be buildings, transmission networks, aircraft, automobiles or military vehicles, trains, naval vessels, and mass transit systems.

Government and military products provide a comprehensive body of knowledge for system modernization and updates. Key references have been developed by the defense industry and can be particular to their needs.

Key factors and questions that must be considered by the systems engineer when making modifications and upgrades to a product or service include

- type of system (space, air, ground, maritime, and safety critical)
- missions and scenarios of expected operational usage
- policy and legal requirements that are imposed by certain agencies or business markets
- product or service LCC's
- electromagnetic spectrum usage expected, including change in RF emissions
- system original equipment manufacturer (OEM) and key suppliers, and availability of parts and subsystems
- understanding and documenting the functions, interfaces, and performance requirements, including environmental testing and validation
- system integration challenges posed by the prevalence of system-of-systems solutions and corresponding interoperability issues between legacy, modified, and new systems
- amount of regression testing to be performed on the existing software

Key processes and procedures that should be considered during product and service modernization include

- legislative policy adherence review and certification
- safety critical review
- engineering change management and configuration control
- analysis of alternatives
- warranty and product return process implementation
- availability of manufacturing and supplier sources and products

Application to Product Systems

Product modernization involves understanding and managing a list of product deficiencies, prioritizing change requests, and handling customer issues associated with product usage. The *INCOSE Systems Engineering Handbook* emphasizes the use of Failure Modes, Effects, and Criticality Analysis (FMECA) to understand the root causes of product failures and provide the basis for making any product changes.

Product modernization uses the engineering change management principle of change control boards to review and implement product changes and improvements. The U.S. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics (OUSD AT&L) provides an online reference for product modernization and the use of an ECP to document planned product or service modernization efforts.

Product modernization and upgrades require the use of system documentation. A key part of the product change process is to change the supporting system documentation functions, interfaces, modes, performance requirements, and limitations. Both INCOSE (2012) and Blanchard and Fabrycky (2005) stress the importance of understanding the intended usage of the product or service documented in the form of a concept of operations.

If system documentation is not available, reverse engineering is required to capture the proper “as is configuration” of the system and to gain understanding of system behavior prior to making any changes. Seacord, Plakosh, and Lewis's *Modernizing Legacy Systems* (2003), explains the importance of documenting the existing architecture of a

system, including documenting the software architecture prior to making any changes. Chapter 5 of Seacord, Plakosh, and Lewis provides a framework for understanding and documenting a legacy system (2003). The authors point out that the product or service software will undergo a transformation during modernization and upgrades. Chapter 5 introduces a horseshoe model that includes functional transformation, code transformation, and architecture transformation (Seacord, Plakosh, and Lewis 2005).

During system verification and validation (after product change), it is important to perform regression testing on the portions of the system that were not modified to confirm that upgrades did not impact the existing functions and behaviors of the system. The degree and amount of regression testing depends on the type of change made to the system and whether the upgrade includes any changes to those functions or interfaces involved with system safety. INCOSE (2012) recommends the use of a requirements verification traceability matrix to assist the systems engineer during regression testing.

It is important to consider changes to the system support environment. Change may require modification or additions to the system test equipment and other support elements such as packaging and transportation.

Some commercial products contain components and subsystems where modernization activities cannot be performed. An example of these types of commercial systems can be seen in looking at consumer electronics, such as radios and computer components. The purchase price of these commercial systems is low enough that upgrades are not economical and are considered cost prohibitive.

Application to Service Systems

Service system modernization may require regulatory changes to allow the use of new technologies and new materials. Service system modernization requires backward compatibility to previous provided service capability during the period of change. Service system modernization also generally spans large geographical areas, requiring a phase-based change and implementation strategy. Transportation systems, such as highways, provide service to many different types of consumers and span large geographical areas. Modernization of transportation systems often requires reverse engineering prior to making changes to understand how traffic monitoring devices such as metering, cameras, and toll tags interface with the rest of the system. The California Department of Transportation's (CDOT's) Systems Engineering Guidebook for Intelligent Transportation Systems (ITS) (2005) adds reverse engineering to the process steps for system upgrade. In addition, this reference points out the need to maintain system integrity and defines integrity to include the accurate documentation of the system's functional, performance, and physical requirements in the form of requirements, design, and support specifications.

Software modernization is discussed in the *Guide to the Software Engineering Body of Knowledge (SWEBOK)* (Bourque and Fairley, 2014).

Application to Enterprises

Enterprise system modernization must consider the location of the modification and the conditions under which the work will be performed. The largest challenge is implementing the changes while the system remains operational. In these cases, disruption of ongoing operations is a serious risk. For some systems, the transition between the old and new configuration is particularly important and must be carefully planned.

Enterprise system modernization may require coordination of changes across international boundaries. Enterprise modifications normally occur at a lower level of the system hierarchy. Change in requirements at the system level would normally constitute a new system or a new model of a system.

The *INCOSE UK Chapter Supplementary Guidance* (2010) discusses the change to the architecture of the system. In cases where a component is added or changed, this change will constitute a change to the architecture. As an example, the global positioning system (GPS) is an enterprise system implemented by the United States military but used by both commercial and government consumers worldwide. Modernization may involve changes to only a

certain segment of the enterprise, such as the ground user segment to reduce size, weight, and power. Modernization may only occur in certain geographical areas of operation. For example, the air transportation system consists of multiple countries and governing bodies dispersed over the entire world. Changes can occur locally or can require coordination and integration world-wide.

Other Topics

The Vee Model for Modifications

Figure 1 below illustrates how the standard Vee model would be applied to a system modification. This Vee model is for the entire system; the key point is that if a modification is being initiated at a lower level of the system hierarchy, the Vee model must be entered at that level as shown in the figure. The figure shows three entry points to the Vee model. As the *INCOSE UK Chapter Supplementary Guidance* (2010) points out, the Vee model may be entered multiple times during the life of the system.

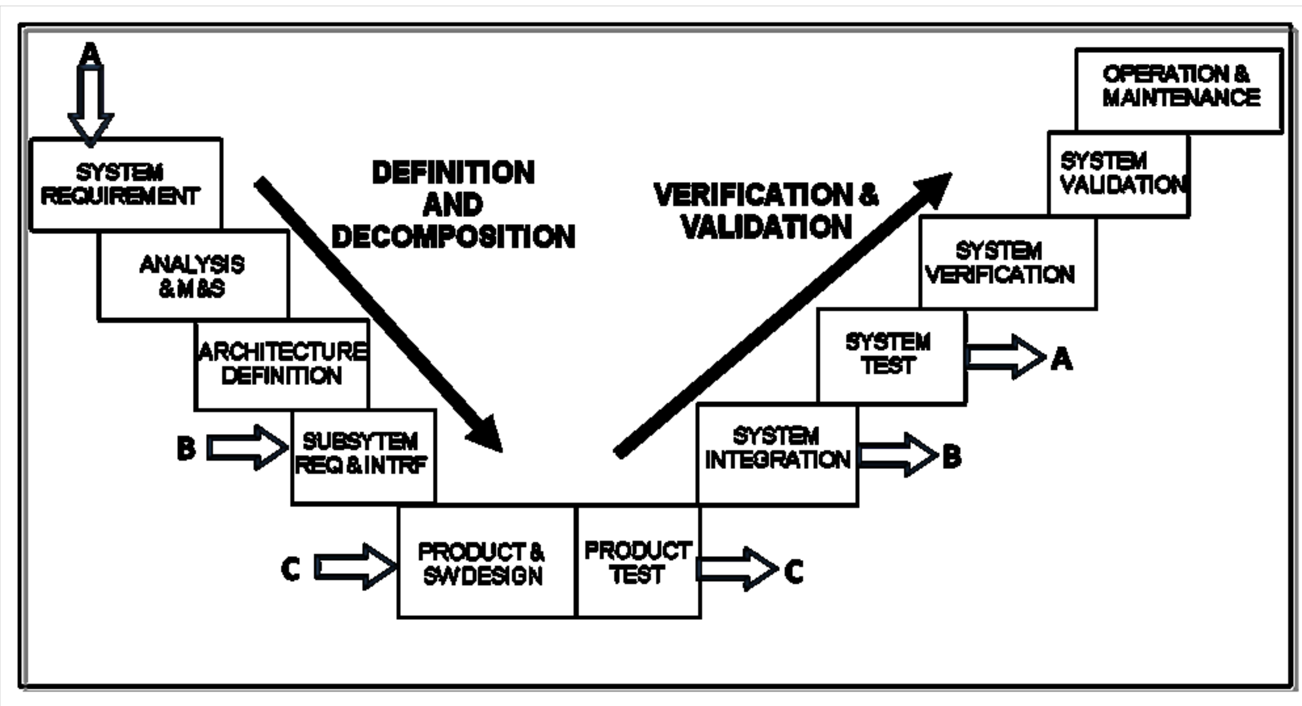


Figure 1. The Vee Model for Modifications at the Three Different Levels. (SEBoK Original)

A change to the system that does not change the system capabilities but does change the requirements and design of a subsystem that may be introduced into the process at point B on the Vee model (see Figure 1). Changes of this type could provide a new subsystem, such as a computer system, that meets the system-level requirements but has differences from the original, which necessitates modifications to the lower-level requirements and design, such as changing disk memory to solid state memory. The process for implementing changes starting at this point has been described by Nguyen (2006). Modification introduced at points B or C (in Figure 1) necessitate flowing the requirements upward through their “parent” requirements to the system-level requirements.

There are many cases where the change to a system needs to be introduced at the lowest levels of the architectural hierarchy; here, the entry point to the process is at point C on the Vee model. These cases are typically related to obsolete parts caused by changes in technology or due to reliability issues with subsystems and parts chosen for the original design. A change at this level should be F3I compatible so that none of the higher-level requirements are affected. The systems engineer must ensure there is no impact at the higher levels; when this does occur, it must be immediately identified and worked out with the customer and the other stakeholders.

In “Life extension of Civil Infrastructural works - a systems engineering approach” vsan der Laan (2008) provides a maintenance process that interacts with the system engineering process, represented by the Vee model. His life extension (or modernization) process model includes reverse engineering to obtain the system definition necessary for the modernization process. Consideration of the total lifecycle of the system will result in the capture of all of the records necessary for later upgrade; however, for many reasons, the systems engineer will find that the necessary information has not been captured or maintained.

Practical Considerations

As pointed out by the *INCOSE UK Chapter Supplementary Guidance* (2010) there may be multiple modifications to a system in its lifetime. Often these modifications occur concurrently. This situation requires special attention and there are two methods for managing it. The first is called the “block” method. This means that a group of systems are in the process of being modified simultaneously and will be deployed together as a group at a specific time. This method is meant to ensure that at the end state, all the modifications have been coordinated and integrated so there are no conflicts and no non-compliance issues with the system-level requirements. The second method is called continuous integration and is meant to occur concurrently with the block method. Information management systems provide an example of a commercial system where multiple changes can occur concurrently. The information management system hardware and network modernization will cause the system software to undergo changes. Software release management is used to coordinate the proper timing for the distribution of system software changes to end-users (Michigan Department of Information Technology, 2008).

Application of Commercial-Off-the-Shelf Components

Currently, a prominent consideration is the use of commercial-off-the-shelf (COTS) components. The application of COTS subsystems, components, and technologies to system life management provides a combination of advantages and risks. The first advantage is the inherent technological advancements that come with COTS components. COTS components continue to evolve toward a higher degree of functional integration. They provide increased functionality, while shrinking in physical size. The other advantage to using COTS components is that they typically have a lower cost.

The risks associated with using COTS during system life management involve component obsolescence and changes to system interfaces. Commercial market forces drive some components to obsolescence within two years or less. Application of COTS requires careful consideration to form factor and interface (physical and electrical).

References

Works Cited

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook,* version 3.2, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methdology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

Nguyen, L. 2006. “Adapting the Vee Model to Accomplish Systems Engineering on Change Projects.” Paper presented at 9th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference, San Diego, CA, USA.

OUSD(AT&L). 2012. “On-line policies, procedures, and planning references.” Office of the Under Secretary of Defense for Acquisition, Transportation and Logistics, US Department of Defense (DoD). Accessed on August 30, 2012. Available at: http://www.acq.osd.mil/log/

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

van der Laan, J. 2008. “Life extension of Civil Infrastructural works - a systems engineering approach.” Proceedings of the 18th annual International Symposium of the International Council on Systems Engineering, Utrecht, the Netherlands.

Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. “Logistics and Materiel Readiness On-line policies, procedures, and planning references.” Arlington, VA, USA: Office of the Under Secretary of Defense for Aquisition, Transportation and Logistics (OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

Additional References

Braunstein, A. 2007. “Balancing Hardware End-of-Life Costs and Responsibilities.” Westport, CT, USA: Experture Group, ETS 07-12-18.

Casetta, E. 2001. *Transportation Systems Engineering: Theory and methods*. New York, NY, USA: Kluwer Publishers Academic, Springer.

DAU. 2010. “Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge.” In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

FAA. 2006. "Section 4.1" in “Systems Engineering Manual.” Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. “Radio and Television Broadcast Rules.” Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

IEC. 2007. *Obsolescence Management - Application Guide*, ed 1.0. Geneva, Switzerland: International Electrotechnical Commission, IEC 62302.

IEEE. 2010. *IEEE Standard Framework for Reliability Prediction of Hardware*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1413.

IEEE. 1998. *IEEE Standard Reliability Program for the Development and Production of Electronic Systems and Equipment*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1332.

IEEE. 2008. *IEEE Recommended practice on Software Reliability*. New York: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1633.

IEEE. 2005. *IEEE Standard for Software Configuration Management Plans*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 828.

INCOSE. 2010. “In-service systems working group.” San Diego, CA, USA: International Council on Systems Engineering (INCOSE).

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook, version 3.2*, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. “Systems Engineering Body of Knowledge, provisional,” version 2.0. Singapore.

ISO/IEC. 2003. “Industrial Automation Systems Integration-Integration of Life-Cycle Data for Process Plants including Oil, Gas Production Factilies.” Geneva, Switzerland: International Organization for Standardization (ISO)/International Electro technical Commission (IEC).

Jackson, S. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Design and Process Science.* 11(2): 91-108, 110.

Jackson, S. 1997. *Systems Engineering for Commercial Aircraft.* Surrey, UK: Ashgate Publishing, Ltd.

Koopman, P. 1999. “Life Cycle Considerations.” In Carnegie-Mellon University (CMU). Pittsburgh, PA, USA, accessed August 5, 2010. Available at: http://www.ece.cmu.edu/~koopman/des_s99/life_cycle/index.html.

Livingston, H. 2010. “GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices.” McClellan, CA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methdology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C.: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Nguyen, L. 2006. “Adapting the Vee Model to Accomplish Systems Engineering on Change Projects.” Paper presented at 9th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference, San Diego, CA, USA.

Reason, J. 1997. *Managing the Risks of Organizational Accident.* Aldershot, UK: Ashgate.

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transpofration (NDDOT).

SAE International. 2010. “Standards: Automotive--Maintenance and Aftermarket.” Warrendale, PA: Society of Automotive Engineers (SAE) International.

Schafer, D.L. 2003. “Keeping Pace With Technology Advances When Funding Resources Are Diminished.” Paper presented at Auto Test Con. IEEE Systems Readiness Technology Conference, Anaheim, CA, USA: 584.

SEI. 2010. “Software Engineering Institute.” In Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU). Pittsburgh, PA, accessed August 5, 2010. http://www.sei.cmu.edu.

SOLE. 2009. “Applications Divisons.” In The International Society of Logistics (SOLE). Hyattsville, MD, USA, accessed August 5, 2010. http://www.sole.org/appdiv.asp.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Disposal and Retirement

Product or service disposal and retirement is an important part of system life management. At some point, any deployed system will become one of the following: uneconomical to maintain; obsolete; or unrepairable. A comprehensive systems engineering process includes an anticipated equipment phase-out period and takes disposal into account in the design and life cycle cost assessment. (See other knowledge areas in Part 3 for discussion on life cycle metrics and assessment.)

A public focus on sustaining a clean environment encourages contemporary systems engineering (SE) design to consider recycling, reuse, and responsible disposal techniques. (See Environmental Engineering for additional discussion.)

Topic Overview

According to the INCOSE Systems Engineering Handbook (2012), “The purpose of the disposal process is to remove a system element from the operation environment with the intent of permanently terminating its use; and to deal with any hazardous or toxic materials or waste products in accordance with the applicable guidance, policy, regulation, and statutes." In addition to technological and economical factors, the system-of-interest (SoI) must be compatible, acceptable, and ultimately address the design of a system for the environment in terms of ecological, political, and social considerations.

Ecological considerations associated with system disposal or retirement are of prime importance. The most concerning problems associated with waste management include

- Air Pollution and Control,
- Water Pollution and Control,
- Noise Pollution and Control,
- Radiation, and

- Solid Waste.

In the US, the Environmental Protection Agency (EPA) and the Occupational Safety and Health Administration (OSHA) govern disposal and retirement of commercial systems. Similar organizations perform this function in other countries. OSHA addresses hazardous materials under the 1910-119A List of Highly Hazardous Chemicals, Toxics, and Reactives (OSHA 2010). System disposal and retirement spans both commercial and government developed products and services. While both the commercial and government sectors have common goals, methods differ during the disposition of materials associated with military systems.

US DoD Directive 4160.21-M, *Defense Material Disposition Manual* (1997) outlines the requirements of the Federal Property Management Regulation (FPMR) and other laws and regulations as appropriate regarding the disposition of excess, surplus, and foreign excess personal property (FEPP). Military system disposal activities must be compliant with EPA and OSHA requirements.

Application to Product Systems

Product system retirement may include system disposal activities or preservation activities (e.g., mothballing) if there is a chance the system may be called upon for use at a later time.

Systems Engineering and Analysis has several chapters that discuss the topics of design for goals such as “green engineering,” reliability, maintainability, logistics, supportability, producibility, disposability, and sustainability. Chapter 16 provides a succinct discussion of green engineering considerations and ecology-based manufacturing. Chapter 17 discusses life cycle costing and the inclusion of system disposal and retirement costs (Blanchard and Fabrycky 2011).

Some disposal of a system's components occurs during the system’s operational life. This happens when the components fail and are replaced. As a result, the tasks and resources needed to remove them from the system need to be planned well before a demand for disposal exists.

Transportation of failed items, handling equipment, special training requirements for personnel, facilities, technical procedures, technical documentation updates, hazardous material (HAZMAT) remediation, all associated costs, and reclamation or salvage value for precious metals and recyclable components are important considerations during system planning. Phase-out and disposal planning addresses when disposal should take place, the economic feasibility of the disposal methods used, and what the effects on the inventory and support infrastructure, safety, environmental requirements, and impact to the environment will be (Blanchard 2010).

Disposal is the least efficient and least desirable alternative for the processing of waste material (Finlayson and Herdlick 2008).

The EPA collects information regarding the generation, management and final disposition of hazardous wastes regulated under the Resource Conservation and Recovery Act of 1976 (RCRA). EPA waste management regulations are codified at 40 C.F.R. parts 239-282. Regulations regarding management of hazardous wastes begin at 40 C.F.R. part 260. Most states have enacted laws and promulgated regulations that are at least as stringent as federal regulations.

Due to the extensive tracking of the life of hazardous waste, the overall process has become known as the "cradle-to-grave system". Stringent bookkeeping and reporting requirements have been levied on generators, transporters, and operators of treatment, storage, and disposal facilities that handle hazardous waste.

Unfortunately, disposability has a lower priority compared to other activities associated with product development. This is due to the fact that typically, the disposal process is viewed as an external activity to the entity that is in custody of the system at the time. Reasons behind this view include:

- There is no direct revenue associated with the disposal process and the majority of the cost associated with the disposal process is initially hidden.

- Typically, someone outside of SE performs the disposal activities. For example, neither a car manufacturer nor the car's first buyer may be concerned about a car's disposal since the car will usually be sold before disposal.

The European Union’s Registration, Evaluation, Authorization, and Restriction of Chemicals (REACH) regulation requires manufacturers and importers of chemicals and products to register and disclose substances in products when specific thresholds and criteria are met (European Parliament 2007). The European Chemicals Agency (ECHA) manages REACH processes. Numerous substances will be added to the list of substances already restricted under European legislation; a new regulation emerged when the Restriction on Hazardous Substances (RoHS) in electrical and electronic equipment was adopted in 2003.

Requirements for substance use and availability are changing across the globe. Identifying the use of materials in the supply chain that may face restriction is an important part of system life management. System disposal and retirement requires upfront planning and the development of a disposal plan to manage the activities. An important consideration during system retirement is the proper planning required to update the facilities needed to support the system during retirement, as explained in the *California Department of Transportation Systems Engineering Guidebook* (2005).

Disposal needs to take into account environmental and personal risks associated with the decommissioning of a system and all hazardous materials need to be accounted for. The decommissioning of a nuclear power plant is a prime example of hazardous material control and exemplifies the need for properly handling and transporting residual materials resulting from the retirement of certain systems.

The US Defense Logistics Agency (DLA) is the lead military agency responsible for providing guidance for worldwide reuse, recycling, and disposal of military products. A critical responsibility of the military services and defense agencies is demilitarization prior to disposal.

Application to Service Systems

An important consideration during service system retirement or disposal is the proper continuation of services for the consumers of the system. As an existing service system is decommissioned, a plan should be adopted to bring new systems online that operate in parallel of the existing system so that service interruption is kept to a minimum. This parallel operation needs to be carefully scheduled and can occur over a significant period of time.

Examples of parallel operation include phasing-in new Air Traffic Control (ATC) systems (FAA 2006), the migration from analog television to new digital television modulation (FCC 2009), the transition to Internet protocol version 6 (IPv6), maintaining water handling systems, and maintaining large commercial transportation systems, such as rail and shipping vessels.

The *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)* provides planning guidance for the retirement and replacement of large transportation systems. Chapter 4.7 identifies several factors which can shorten the useful life of a transportation system and lead to early retirement, such as the lack of proper documentation, the lack of effective configuration management processes, and the lack of an adequate operations and maintenance budget (Caltrans, and USDOT 2005).

Application to Enterprises

The disposal and retirement of large enterprise systems requires a phased approach, with capital planning being implemented in stages. As in the case of service systems, an enterprise system's disposal and retirement require parallel operation of the replacement system along with the existing (older) system to prevent loss of functionality for the user.

Other Topics

See the OSHA standard (1996) and EPA (2010) website for references that provide listings of hazardous materials. See the DLA Disposal Services website [1] for disposal services sites and additional information on hazardous materials.

Practical Considerations

A prime objective of systems engineering is to design a product or service such that its components can be recycled after the system has been retired. The recycling process should not cause any detrimental effects to the environment.

One of the latest movements in the industry is green engineering. According to the EPA, green engineering is the design, commercialization, and use of processes and products that are technically and economically feasible while minimizing

- the generation of pollutants at the source; and
- the risks to human health and the environment.

See Environmental Engineering for additional information.

References

Works Cited

Blanchard, B. S. 2010. *Logistics Engineering and Management*, 5th ed. Englewood Cliffs, NJ: Prentice Hall, 341-342.

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DoD. 1997. *Defense Materiel Disposition Manual.* Arlington, VA, USA: US Department of Defense, DoD 4160.21 -M.

DLA. 2010. "Defense logistics agency disposition services." In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla. mil.

ECHA. 2010. "European Chemicals Agency (ECHA)." In European Chemicals Agency (ECHA). Helsinki, Finland. Available at: http://echa.europa.edu/home_en.asp.

EPA. 2010. "Wastes In U.S. Environmental Protection Agency (EPA)." Washington, D.C. Available at: http:// www.epa.gov/epawaste/index.htm.

European Parliament. 2007. Regulation (EC) no 1907/2006 of the european parliament and of the council of 18 december 2006 concerning the registration, evaluation, authorisation and restriction of chemicals (REACH), establishing a european chemicals agency, amending directive 1999/45/EC and repealing council regulation (EEC)

no 793/93 and commission regulation (EC) no 1488/94 as well as council directive 76/769/EEC and commission directives 91/155/EEC, 93/67/EEC, 93/105/EC and 2000/21/EC. Official Journal of the European Union 29 (5): 136/3,136/280.

FAA. 2006. "Section 4.1" in "Systems Engineering Manual." Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. "Radio and Television Broadcast Rules." Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

OSHA. 1996. "Hazardous Materials: Appendix A: List of Highly Hazardous Chemicals, Toxics and Reactives." Washington, DC, USA: Occupational Safety and Health Administration (OSHA)/U.S. Department of Labor (DoL), 1910.119(a).

Resource Conservation and Recovery Act of 1976, 42 U.S.C. §6901 et seq. (1976)

Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson, S. 2007. "A Multidisciplinary Framework for Resilience to Disasters and Disruptions." *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. "Logistics and Materiel Readiness On-line policies, procedures, and planning references." Arlington, VA, USA: Office of the Under Secretary of Defense for Aquisition, Transportation and Logistics (OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

Additional References

Blanchard, B. S. 2010. *Logistics Engineering and Management*, 5th ed. Englewood Cliffs, NJ: Prentice Hall, 341-342.

Casetta, E. 2001. *Transportation Systems Engineering: Theory and methods*. New York, NY, USA: Kluwer Publishers Academic, Springer.

DAU. 2010. "Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge." In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

DLA. 2010. "Defense logistics agency disposition services." In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla.mil.

ECHA. 2010. "European Chemicals Agency (ECHA)." In European Chemicals Agency (ECHA). Helsinki, Finland. Available at: http://echa.europa.edu/home_en.asp.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

EPA. 2010. “Wastes In U.S. Environmental Protection Agency (EPA)." Washington, D.C. Available at: http:// www.epa.gov/epawaste/index.htm.

European Parliament. 2007. Regulation (EC) no 1907/2006 of the european parliament and of the council of 18 december 2006 concerning the registration, evaluation, authorisation and restriction of chemicals (REACH), establishing a european chemicals agency, amending directive 1999/45/EC and repealing council regulation (EEC) no 793/93 and commission regulation (EC) no 1488/94 as well as council directive 76/769/EEC and commission directives 91/155/EEC, 93/67/EEC, 93/105/EC and 2000/21/EC. Official Journal of the European Union 29 (5): 136/3,136/280.

FAA. 2006. "Section 4.1" in “Systems Engineering Manual.” Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. “Radio and Television Broadcast Rules.” Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

FSA. 2010. “Template for 'System Retirement Plan' and 'System Disposal Plan'.” In Federal Student Aid (FSA)/U.S. Department of Eduation (DoEd). Washington, DC, USA. Accessed August 5, 2010. Available at: http:// federalstudentaid.ed.gov/business/lcm.html.

IEEE 2005. *IEEE Standard for Software Configuration Management Plans*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 828.

Ihii, K., C.F. Eubanks, and P. Di Marco. 1994. “Design for Product Retirement and Material Life-Cycle.” *Materials & Design.* 15(4): 225-33.

INCOSE. 2010. “In-service systems working group.” San Diego, CA, USA: International Council on Systems Engineering (INCOSE).

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook, version 3.2*, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. “Systems Engineering Body of Knowledge, provisional,” version 2.0. Singapore: Institute of Engineers Singapore.

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methdology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

Minneapolis-St. Paul Chapter of SOLE. 2003. "Systems Engineering in Systems Deployment and Retirement, presented to INCOSE." Minneapolis-St. Paul, MN, USA: International Society of Logistics (SOLE), Minneapolis-St. Paul Chapter.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C.: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

OSHA. 1996. “Hazardous Materials: Appendix A: List of Highly Hazardous Chemicals, Toxics and Reactives.” Washington, DC, USA: Occupational Safety and Health Administration (OSHA)/U.S. Department of Labor (DoL),

1910.119(a).

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transpofration (NDDOT).

SAE International. 2010. “Standards: Automotive--Maintenance and Aftermarket.” Warrendale, PA: Society of Automotive Engineers (SAE) International.

Schafer, D.L. 2003. “Keeping Pace With Technology Advances When Funding Resources Are Diminished.” Paper presented at Auto Test Con. IEEE Systems Readiness Technology Conference, Anaheim, CA, USA: 584.

SOLE. 2009. “Applications Divisons.” In The International Society of Logistics (SOLE). Hyattsville, MD, USA, accessed August 5, 2010. http://www.sole.org/appdiv.asp.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.dtc.dla.mil

Knowledge Area: Systems Engineering Standards

Systems Engineering Standards

This knowledge area (KA) focuses on the standards and technical protocols that are relevant to systems engineering. It looks at the types of standards, some of the key standards, and the alignment efforts to achieve a consistent set of standards. It then compares some of the standards, and surveys the application of the standards. Note that many of these standards have been used as references throughout Part 3.

Topics

Each part of the SEBoK is divided into KA's, which are groupings of information with a related theme. The KA's in turn are divided into topics. This KA contains the following topics:

- Relevant Standards
- Alignment and Comparison of the Standards
- Application of Systems Engineering Standards

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Relevant Standards

There are a multitude of standards across a number of standards development organizations (SDOs) that are related to systems engineering and systems domains. This topic examines the types of standards and provides a summary of the relevant standards for systems engineering (SE).

Standards Taxonomies and Types of Standards

There are many types of standards that focus on different aspects of SE. Thus, it can be helpful to have a taxonomy that classifies the types of standards and the objective of each type. Table 1 provides the types of the current standards and a description of the types. Refer to the Modeling Standards for a list of relevant system modeling standards.

Table 1. Types of Systems Engineering Standards. (SEBoK Original)

Standard Type	Description of Type
Concepts and Terminology	• Defines the terminology and describes the concepts of a specific domain.
Process	• Elaborates a specific process, giving normative requirements for the essential elements of the process. It may give guidance to the requirements.
Requirements	• Describes the requirements for something. • Most often used for actions, activities, or practices and not objects (see specifications).
Procedure (Practice, Activity)	• A specific procedure. Instructions or requirements on how to do something. • Could be a description of best practices. • Sometimes guidance and sometimes normative.
Guidance	• Usually an interpretation and guidance of a published standard.
Management System	• Requirements for management.
Specification	• Specifies the form, attributes, or properties of a subject artifact. • Usually an object and usually normative.
Reference Model	• A reference model or collection of specifications of which a reference model is composed.
Process Reference Model (PRM)	• A collection of processes necessary and sufficient to achieve a nominated business outcome.
Process Assessment Model (PAM)	• Requirements and guidance for assessing attributes of nominated processes or attributes of a nominated collection of processes.
Guide to Body of Knowledge (BOK)	• Collects and describes the current body of knowledge in a domain, or guidance to the body of knowledge.

Systems Engineering Related Standards

Summary of Systems Engineering Related Standards

Table 2 contains a summary of SE related standards. This table does not include all SE related standards, as there are many are focused on a specific domain, sector, or user group (e.g., it does not include standards from a specific government agenda). The table does include standards that are considered to be widely applicable systems engineering and systems life cycle management system life cycle processes, such as ISO/IEC/IEEE 15288 (2015). Where available, there is a link to the official abstract for the standard.

Table 2. Summary of Systems Engineering Standards. (SEBoK Original)

Document ID	Document Title	Organization
ISO/IEC/IEEE 15288 [1]	Systems and Software Engineering - System Life Cycle Processes	ISO/IEC/IEEE
ISO/IEC/IEEE 24765 [2]	Systems and Software Engineering - Systems and Software Engineering Vocabulary	ISO/IEC/IEEE
ISO/IEC/IEEE 42010 [3]	Systems and Software Engineering - Architecture Description	ISO/IEC/IEEE
ISO/IEC 26702 [4] / IEEE 1220 [5]	Management of the Systems Engineering Process	ISO/IEC/IEEE
ISO/IEC/IEEE 29148 [6]	Systems and Software Engineering - Requirements Engineering	ISO/IEC/IEEE
ISO/IEC/IEEE 16085 [7]	Systems and Software Engineering - Risk Management	ISO/IEC/IEEE
ISO/IEC/IEEE 15939 [8]	Systems and Software Engineering - Measurement Process	ISO/IEC/IEEE
ISO/IEC/IEEE 16326 [9]	Systems and Software Engineering - Project Management	ISO/IEC/IEEE
prEN9277 [10]	Programme management - Guide for the management of Systems Engineering	CEN
EIA 632 [11]	Engineering of a System	TechAmerica
ISO 9001:2008 [12]	Quality Management Systems - Requirements	ISO TC 176
EIA-649-B [13]	National Consensus Standard for Configuration Management	TechAmerica
ISO/IEC/IEEE TR 24748-1 [14]	Systems and Software Engineering - Guide to Life Cycle Management	ISO/IEC/IEEE
ISO/IEC/IEEE TR 24748-2 [15]	Systems and Software Engineering - Guide To The Application of ISO/IEC 15288:2008	ISO/IEC/IEEE
ISO/IEC/IEEE CD 24748-4 [16]	Systems and Software Engineering - Application and management of the systems engineering process	ISO/IEC/IEEE
ISO/IEC DTR 16337 [17]	Systems Engineering Handbook (INCOSE)	ISO/IEC/INCOSE
ISO/IEC/IEEE 15289:2011 [18]	Systems and Software Engineering - Content of Life-Cycle Information Products (Documentation)	ISO/IEC/IEEE
ISO/IEC/IEEE 15026-1:2010 [19]	Systems and Software Engineering - System and Software Assurance – Part 1: Concepts And Vocabulary	ISO/IEC/IEEE
ISO/IEC/IEEE 15026-2:2011 [20]	Systems and Software Engineering - System and Software Assurance – Part 2: Assurance Case	ISO/IEC/IEEE
ISO/IEC/IEEE 15026-3:2011 [21]	Systems and Software Engineering - System and Software Assurance – Part 3: Integrity Levels	ISO/IEC/IEEE
ISO/IEC/IEEE 15026-4:2012 [22]	Systems and Software Engineering - System And Software Assurance – Part 4: Assurance in the Life Cycle	ISO/IEC/IEEE JTC 1
ISO/IEC TR 90005:2008 [23]	Guidelines for the Application of ISO 9001 to Systems Life Cycle Processes	ISO/IEC JTC 1
ISO 10303-233:2012 [24]	Systems Engineering Data Interchange Standard	ISO TC 184
ECSS-E-ST-10C [25]	Systems Engineering General Requirements	ECSS
ECSS-E-ST-10-02 [26]	Space Engineering - Verification {Note - standard is canceled}	ECSS
ECSS-E-ST-10-06 [27]	Space Engineering - Technical Requirements Specification	ECSS
ECSS-E-ST-10-24 [28]	Space Engineering - Interface Control	ECSS

ECSS-M-ST-10 [29]	Space Project Management - Project Planning and Implementation	ECSS
ECSS-M-ST-40 [30]	Space Project Management - Configuration and Information Management	ECSS
ECSS-M-00-03 [31]	Space Project Management - Risk Management	ECSS
ISO 31000:2009 [32]	Risk Management - Principles and Guidelines	ISO
ISO 31010:2009 [33]	Risk Management - Risk Assessment Techniques	ISO
ISO 19439:2006 [34]	Enterprise Integration - Framework for Enterprise Modeling	ISO
ISO 15704:2000 [35]	Requirements for Enterprise - Reference Architectures and Methodologies	ISO
EIA 748 [36]	Earned Value Management System	TechAmerica

Breadth and Level of Detail of Key Systems Engineering Related Standards

Figure 1 shows the level of detail and the coverage of the life cycle for some key standards or groups of standards.

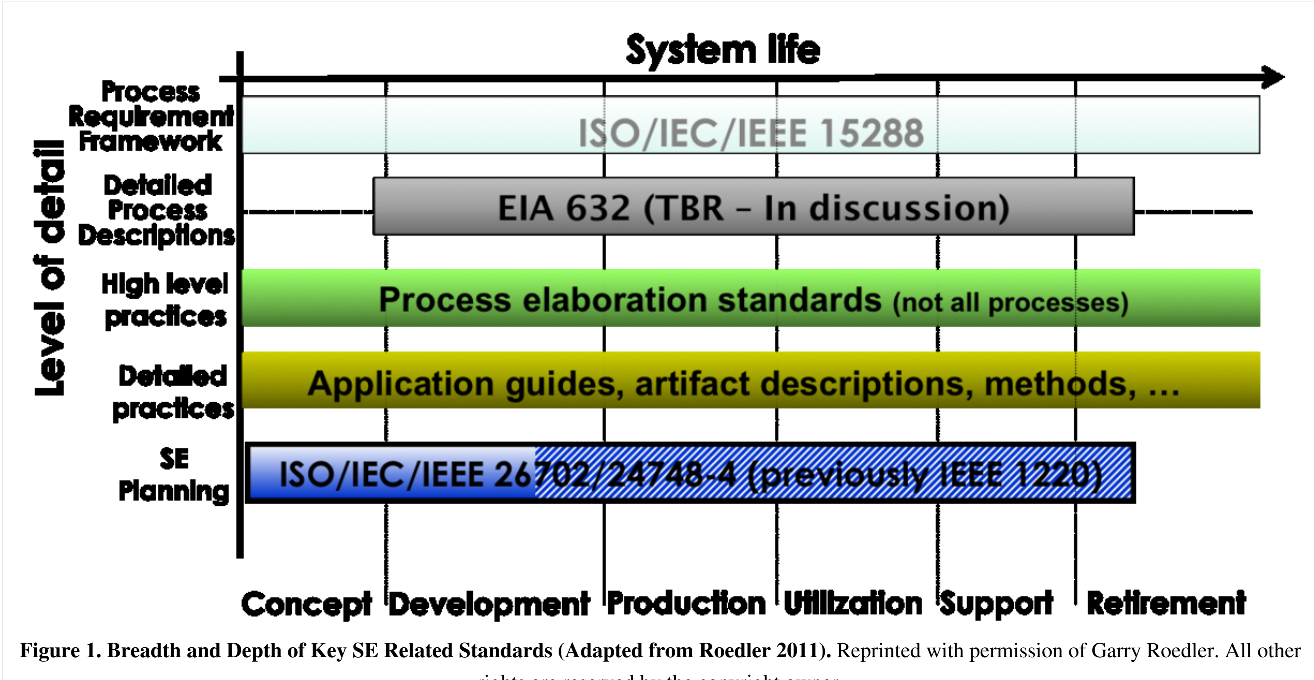


Figure 1. Breadth and Depth of Key SE Related Standards (Adapted from Roedler 2011). Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

Practical Considerations

Key pitfalls and good practices related to systems engineering standards are described in the next two sections.

Pitfalls

Some of the key pitfalls encountered in the selection and use of SE standards are provided in Table 3.

Table 3. Pitfalls in Using Systems Engineering Standards. (SEBoK Original)

Pitfall Name	Pitfall Description
Turnkey Process Provision	• Expecting the standard to fully provide your SE processes without any elaboration or tailoring.
No Need for Knowledge	• Expecting that the standard can be used without any functional or domain knowledge since the standard is the product of collective industry knowledge.
No Process Integration	• Lack of integrating the standards requirements with the organization or project processes.

Good Practices

Some good practices as gathered from the references and provided in Table 4.

Table 4. Good Practices in Using Systems Engineering Standards. (SEBoK Original)

Good Practice Name	Good Practice Description
Tailor for Business Needs	• Tailor the standard within conformance requirements to best meet business needs.
Integration into Project	• Requirements of the standard should be integrated into the project via processes or product/service requirements.

References

Works Cited

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes." Proceedings of the 4th Asian Pacific Council on Systems Engineering (APCOSE) Conference, 4-6 October 2010, Keelung, Taiwan.

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

Primary References

ANSI/EIA. 2003. *Processes for Engineering a System.* Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Engineering Vocabulary.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 24765:2009.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements engineering.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

Roedler, G. 2010. *An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes.* Proceedings of the 4th Asian Pacific Council on Systems Engineering (APCOSE) Conference, 4-6 October 2010, Keelung, Taiwan.

Additional References

ISO. 2003. *Space Systems - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 17666:2003.

ISO. 2009. *Risk Management - Principles and Guidelines.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 31000:2009.

ISO/IEC. 2009. *Risk Management - Risk Assessment Techniques.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 31010:2009.

ISO/IEC/IEEE. 2006. *Systems and Software Engineering - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16085.

ISO/IEC/IEEE. 2007. *Systems and Software Engineering - Measurement Process.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 15939.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Project Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 16326.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Assurance, Part 1: Concepts and definitions.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-1.

ISO/IEC/IEEE. 2010. *Systems and Software Engineering - System and Software Assurance, Part 2: Assurance case.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-2.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Content of Life-Cycle Information Products (documentation).* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15289.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - System and Software Assurance, Part 3: Integrity Levels.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-3.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43564
[2] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50518
[3] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=50508
[4] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43693
[5] http://standards.ieee.org/findstds/standard/1220-2005.html
[6] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45171
[7] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40723
[8] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44344
[9] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41977
[10] http://infostore.saiglobal.com/store/details.aspx?ProductID=1615031
[11] http://www.techstreet.com/products/1145585
[12] http://www.iso.org/iso/catalogue_detail?csnumber=46486
[13] http://www.techstreet.com/products/1800866
[14] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50502
[15] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=54994

[16] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=56887
[17] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56186
[18] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54388
[19] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50520
[20] http://www.iso.org/iso/catalogue_detail.htm?csnumber=52926
[21] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57107
[22] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=59927
[23] http://www.iso.org/iso/catalogue_detail?csnumber=41553
[24] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=55257
[25] http://www.inpe.br/twiki/pub/Main/GerenciamentoProjetosEspaciais/ECSS-E-ST-10C(6March2009).pdf
[26] http://www.everyspec.com/ESA/ECSS-E-10-02A_14991/
[27] http://www.inpe.br/twiki/pub/Main/GerenciamentoProjetosEspaciais/ECSS-E-ST-10-06C6March2009.pdf
[28] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCsQFjAA&url=http%3A%2F%2Fwww.ecss.nl%2Fforums%2Fecss%2Fdispatch.cgi%2Fhome%2FshowFile%2F100807%2Fd20130924080101%2FNo%2FECSS-E-ST-10-24C_DIR1(24September2013).doc&ei=Ji5cUrt0woLbBZ2qgOgN&usg=AFQjCNHNB_u3X71aMcFAeiiN2PAZuxGGmQ&bvm=bv.53899372,d.b2I
[29] http://www.everyspec.com/ESA/ECSS-M-ST-10C_REV-1_47763/
[30] http://www.ecss.nl/forums/ecss/dispatch.cgi/standards/showFile/100665/d20080802121136/No/ECSS-M-ST-80C(31July2008).doc
[31] http://www.everyspec.com/ESA/ecss-m-00-03a_2569/
[32] http://www.iso.org/iso/catalogue_detail?csnumber=43170
[33] http://www.iso.org/iso/catalogue_detail?csnumber=51073
[34] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=33833
[35] http://www.iso.org/iso/catalogue_detail.htm?csnumber=28777
[36] http://www.techstreet.com/products/1854970

Alignment and Comparison of the Standards

Over the past decade, a number of the standards development organizations (SDOs) and other industry associations have been working collaboratively to align the systems engineering (SE) and software engineering (SwE) standards. The objective is to have a set of standards that can easily be used concurrently within both engineering disciplines, due to the disparity that often lies within their use of common terminology and concepts.

Problem

There has been a lack of integration both within and across SDOs. This has led to SE and SwE standards that use different terminology, process sets, process structures, levels of prescription, and audiences. These differences have been both between systems and software, and to some extent, within each. The problem has been exacerbated, in whole or part, by competing standards (Roedler 2010).

Cause

The cause of this problem includes several factors, as follows (Roedler 2010):

- culture - “we’re different”, “not invented here” , etc.
- organizational - different teams, committees, etc.
- competition - many SDO's
- domains - focused, narrow view often doesn’t look beyond the domain for commonality

Impact

The impact of this problem includes the following (Roedler 2010):

- Less effective or efficient processes that are not focused on leveraging commonalities. This causes redundancy and has resulted in incompatibilities and inconsistencies between the standards making it difficult to concurrently use them together.
- Less effective solutions that are not focused on a common approach to solve a problem or need.
- Obstacle for communicating (at all levels), working in integrated teams, and leveraging resources.
- Stove-piping due to the incompatibilities, inconsistencies, and lack of leveraging commonalities.

Objective of Alignment

The objective is to make the standards more usable together by achieving the following (Roedler 2010):

- common vocabulary
- single, integrated process set
- single process structure
- jointly planned level of prescription
- suitable across the audiences
- accounts for considerations in a wide range of domains and applications

Alignment of Systems Engineering Standards

Approach

A collaborative effort has been in place for the past decade that includes ISO/IEC JTC1/SC7 (Information Technology, Systems and Software Engineering), the IEEE Computer Society, the International Council on Systems Engineering (INCOSE), and others. A collaborative process is being used to align the standards. This process is built around a foundational set of vocabulary, process definition conventions, and life cycle management concepts provided in ISO/IEC/IEEE 24765 (2009) (*Systems and software engineering vocabulary*), ISO/IEC TR 24774 (2010) (*Guidelines for process description*), and ISO/IEC/IEEE TR 24748-1 (2001) (*Guide to Life Cycle Management*), respectively. At the heart of the approach is the alignment of the ISO/IEC/IEEE 15288 (2015) (system life cycle processes) and ISO/IEC/IEEE 12207 (2008) (*Software life cycle processes*), which provide the top level process framework for life cycle management of systems and software. This enables concurrent and consistent use of the standards to support both systems and software life cycle management on a single project. The approach includes the development or revision of a set of lower level supporting standards and technical reports for elaboration of specific processes, description of practices for specific purposes (e.g., systems/software assurance), description of artifacts, and guidance for the application of the standards.

Past Accomplishments

Significant progress has been made towards the alignment objectives for the groups discussed above. Figure 1 shows a May 2011 snapshot of the status of the standards that are being aligned. In addition, four of the standards shown as "in-process" are complete, but waiting for final publication. The set of standards span ISO/IEC, IEEE, INCOSE, and the Project Management Institute (PMI). This figure depicts the standards in one of many possible taxonomies.

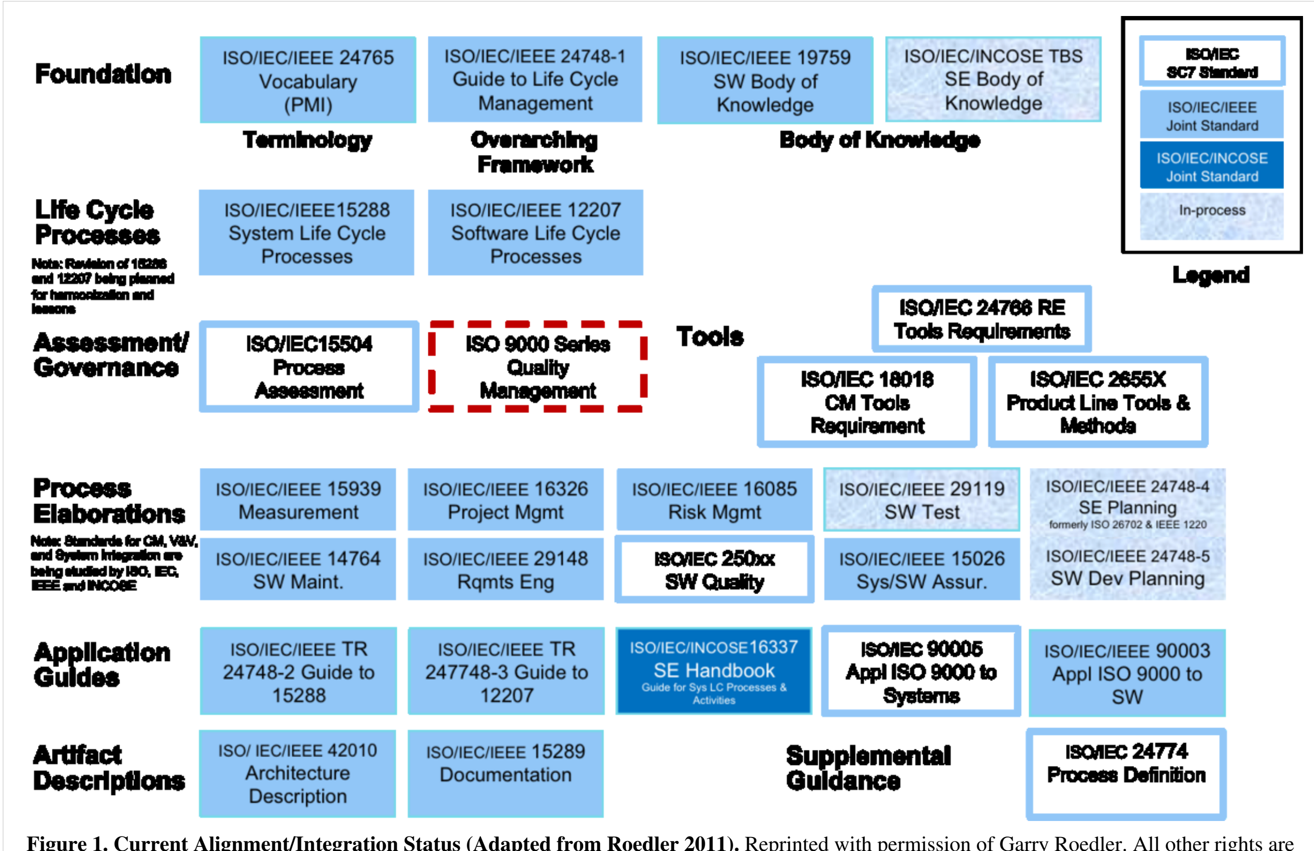


Figure 1. Current Alignment/Integration Status (Adapted from Roedler 2011). Reprinted with permission of Garry Roedler.

Current Efforts

A Life Cycle Process Harmonization Advisory Group has been evaluating the current standards for systems and software engineering. The objective of the group is to provide a set of recommendations for further harmonization of the industry standards. Specifically, its charter includes:

- Performing an architectural analysis and recommend a framework for an integrated set of process standards in software and IT systems domains.
- Making recommendations regarding the future content, structure, and relationships of ISO/IEC 12207 (2008), ISO/IEC 15288 (2015) and their guides, as well as other related SC7 documents.

To support the development of the recommendations, process modeling of ISO/IEC/IEEE 15288 (2015) and ISO/IEC/IEEE 12207 (2008) has been performed and analyzed for consistency, completeness/gaps, and opportunities. In addition, analysis from other working groups, technical liaisons, and users of the standards has been collected. The output of this effort will be a harmonization strategy, set of recommendations for specific standards, and timing/sequencing recommendations (Roedler 2011).

Additionally, as the industry continues to consider harmonization needs of these standards, the collaboration has grown to include the work of the organizations and projects shown in Figure 2. These organizations are working towards the goal of completing a complementary and supplementary set of systems engineering resources that use the same terminology, principles, concepts, practices, and processes and can be used concurrently without issues.

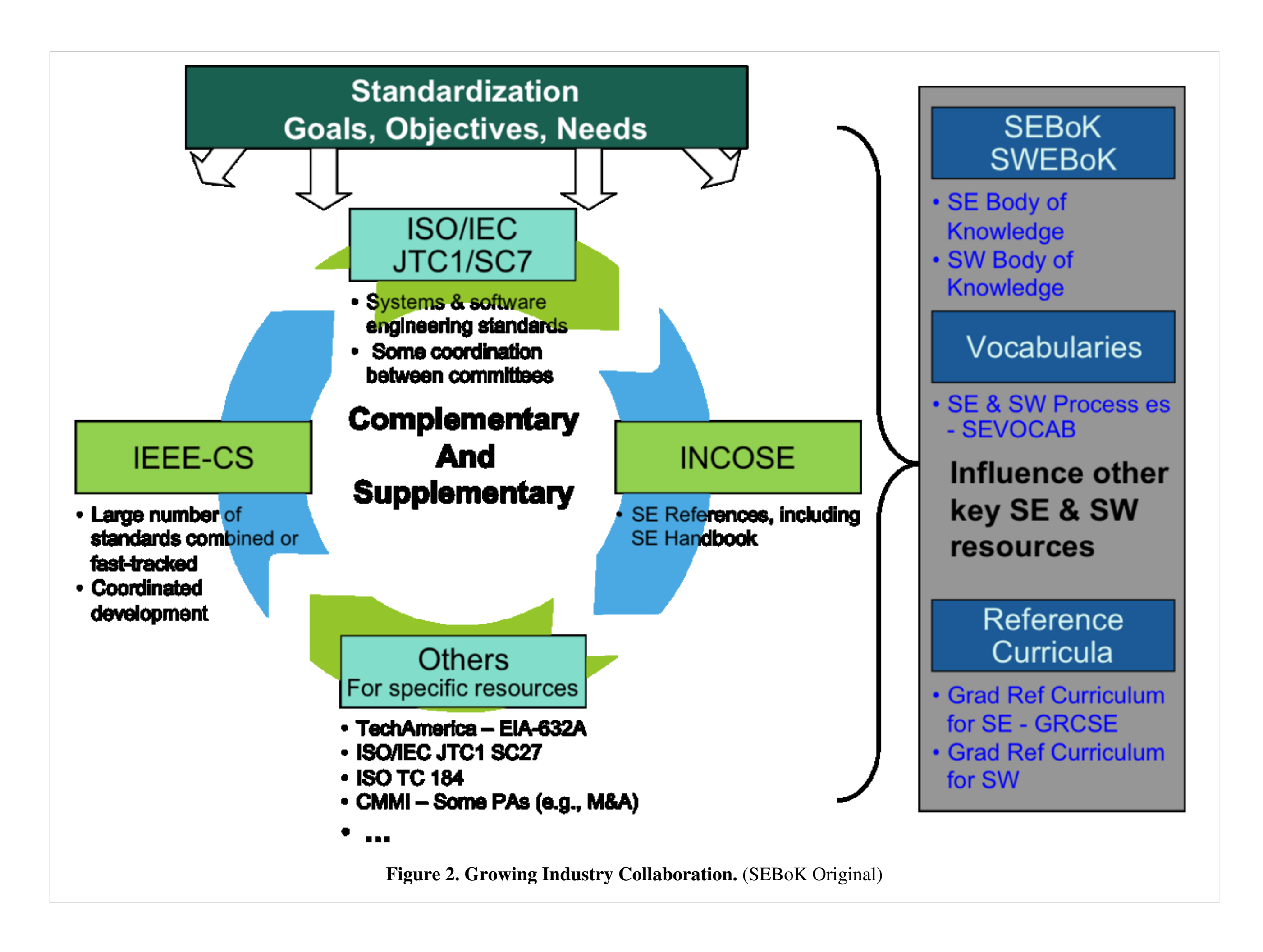


Figure 2. Growing Industry Collaboration. (SEBoK Original)

Comparison of Systems Engineering Standards

See Figure 1 located in the Relevant Standards article to see the breadth and level of detail for many of the SE related standards. Since EIA 632 (2003) (*Engineering of a System*) is currently in revision, a comparison of ISO/IEC/IEEE 15288 (2015) (*System life cycle processes*) and EIA 632 will be deferred until the revision is complete.

Figure 3 shows a comparison of the 3-part technical reports that provide life cycle management guidance. Part 1 is focused on the provision of common terminology and concepts that apply to both systems and software. Part 2 provides guidance that directly supports ISO/IEC/IEEE 15288 (2015) that is specific to systems. And Part 3 provides guidance that directly supports ISO/IEC/IEEE 12207 (2008) that is specific to software (Roedler 2010).

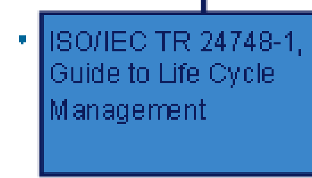


Figure 3. Standards Alignment Results as of May 2011 (Roedler 2011). Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

Practical Considerations

Key pitfalls and good practices related to systems engineering standards are described in the Relevant Standards article.

There are also instances in which standards groups for program management, safety, or other disciplines create standards on topics addressed within systems engineering but use different terminology, culture, etc. One such example is risk management, which has been dealt with by many professional societies from a number of perspectives.

Systems engineers must also be aware of the standards that govern the specialty disciplines that support systems engineering, as discussed in Part 6.

References

Works Cited

ANSI/EIA. 2003. Processes for Engineering a System. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA). ANSI/EIA 632-1998.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC. 2010. *Systems and software engineering -- Life cycle management -- Guidelines for process description*. Geneva, Switzerland: International Organisation for Standardisation/International Electrotechnical Commissions. ISO/IEC TR 24774:2010.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Engineering Vocabulary (SEVocab)*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 24765:2009.

ISO/IEC/IEEE. 2011. "Part 1: Guide for life cycle management," in *Systems and software engineering--life cycle management.*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE TR 24748-1:2010.

ISO/IEEE. 2008. Systems and Software Engineering — Software Life Cycle Processes. Geneva, Switzerland: International Organization for Standards (ISO)/Institute of Electrical & Electronics Engineers (IEEE) Computer Society, ISO/IEEE 12207:2008(E).

Roedler, G. 2010. *An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes.* Asian Pacific Council on Systems Engineering (APCOSE) Conference.

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

Primary References

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes." Asian Pacific Council on Systems Engineering (APCOSE) Conference.

Additional References

ISO. 2003. *Space Systems - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 17666:2003.

ISO. 2009. *Risk Management—Principles and Guidelines.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 31000:2009.

ISO/IEC. 2009. *Risk Management—Risk Assessment Techniques]].* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 31010:2009.

ISO/IEC/IEEE. 2006. *Systems and Software Engineering - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16085.

ISO/IEC/IEEE. 2007. *Systems and Software Engineering - Measurement Process.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 15939.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Project Management]].* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 16326.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Assurance, Part 1: Concepts and definitions.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-1.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Engineering Vocabulary (SEVocab).* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 24765:2009.

ISO/IEC/IEEE. 2010. *Systems and Software Engineering - System and Software Assurance, Part 2: Assurance case.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-2.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Content of Life-Cycle Information Products (documentation).* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15289.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - System and Software Assurance, Part 3: Integrity Levels.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-3.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Application of Systems Engineering Standards

There are many systems engineering standards that have evolved over time, as indicated in Relevant Standards. In particular, there are standards that can have an influence on organizations and their projects as indicated in Figure 1 (below). Some pitfalls and good practices in utilizing standards are also identified in the article on relevant standards. In this article, several additional factors related to the utilization of the standards in systems engineering (SE) are presented.

Standards and their Utilization

A standard is an agreed upon, repeatable way of doing something. It is a published document that contains a technical specification or other precise criteria designed to be used consistently as a rule, guideline, or definition. Standards help to make life simpler and to increase the reliability and the effectiveness of many goods and services we use. Standards are created by bringing together the experience and expertise of all interested parties, such as the producers, sellers, buyers, users, and regulators of a particular material, product, process, or service.

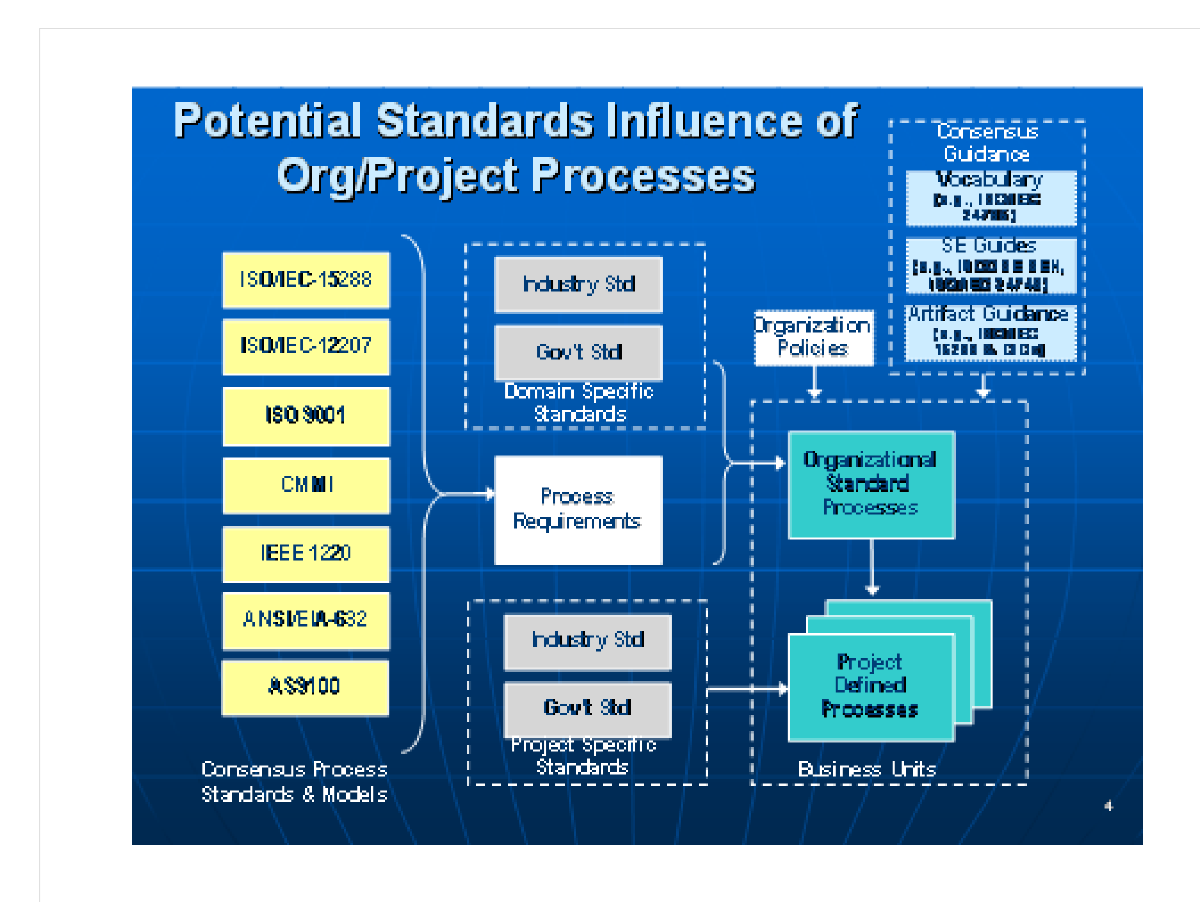


Figure 1. Potential Standards Influence of Organization and Project Processes (Adapted from Roedler 2011). Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

Standards are designed for voluntary use and do not impose any regulations. However, laws and regulations may address certain standards and may make compliance with them compulsory.

Further, organizations and their enterprises may choose to use standards as a means of providing uniformity in their operations and/or the products and services that they produce. The standard becomes a part of the corporate culture. In this regard, it is interesting to note that the ISO/IEC/15288 15288 (2015) standard has provided such guidance and has provided a strong framework for systems engineers as well as systems engineering and business management, as forecast earlier by Arnold and Lawson (2004).

ISO directives [1] state the following:

> *A standard does not in itself impose any obligation upon anyone to follow it. However, such an obligation may be imposed, for example, by legislation or by a contract. In order to be able to claim compliance with a standard, the user (of the standard) needs to be able to identify the requirements he is obliged to satisfy. The user needs also to be able to distinguish these requirements from other provisions where a certain freedom of choice is possible. Clear rules for the use of verbal forms (including modal auxiliaries) are therefore essential.*

Requirements, Recommendations, and Permissions

In order to provide specificity, standards employ verb forms that convey requirements, recommendations, and permissions. For example, the ISO directives specify the following verb usages:

- The word *shall* indicates requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted.
- The word *should* indicates that among several possibilities, one is recommended as particularly suitable without mentioning or excluding others, or that a certain course of action is preferred, but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.
- The word *may* indicates a course of action permissible within the limits of the standard.

The directive also indicates that standards should avoid the use of *will, must,* and other imperatives.

Certification, Conformance, and Compliance

In the context of the management system standards (ISO 9001:2000 and ISO 9001:2008 or ISO 14001:2004), *certification* refers to the issuing of written assurance (the certificate) by an independent external body that it has audited a management system and verified that it conforms to the requirements specified in the standard.

Typically, other more specific systems engineering standards are not the subject of certification. They are self-imposed in order to improve uniformity of organization and enterprise operations or to improve the quality of products and services. Alternatively, they may be dictated by legislation, policy, or as part of a formal agreement between an acquirer and a supplier.

Conformance testing, or type testing, is testing to determine whether a product or system meets some specified standard that has been developed for efficiency or interoperability. To aid in this, many test procedures and test setups have been developed either by the standard's maintainers or by external organizations, such as the Underwriters Laboratory (UL), specifically for testing conformity to standards.

Conformance testing is often performed by external organizations, which is sometimes the standards body itself, to give greater guarantees of compliance. Products tested in such a manner are then advertised as being certified by that external organization as complying with the standard. Service providers, equipment manufacturers, and equipment suppliers rely on this data to ensure quality of service (QoS) through this conformance process.

Tailoring of Standards

Since the SE standards provide guidelines, they are most often tailored to fit the needs of organizations and their enterprises in their operations and/or for the products and services that they provide, as well as to provide agreement in a contract. Tailoring is a process described in an annex to the ISO/IEC/IEEE 15288 (2015) standard.

The ISO/IEC/IEEE 15288 (2015) addresses the issues of conformance, compliance, and tailoring as follows:

- Full conformance, or a claim of full conformance first declares the set of processes for which conformance is claimed. Full conformance is achieved by demonstrating that all of the requirements of the declared set of processes have been satisfied using the outcomes as evidence.
- Tailored conformance is an international standard that used as a basis for establishing a set of processes that do not qualify for full conformance; the clauses of this international standard are selected or modified in accordance with the tailoring process.
- The tailored text, for which tailored conformance is claimed, is declared. Tailored conformance is achieved by demonstrating that requirements for the processes, as tailored, have been satisfied using the outcomes as evidence.
- When the standard is used to help develop an agreement between an acquirer and a supplier, clauses of the standard can be selected for incorporation in the agreement with or without modification. In this case, it is more appropriate for the acquirer and supplier to claim compliance with the agreement than conformance with the

standard.

- Any organization (e.g., a national organization, industrial association, or company) imposing the standard as a condition of trade should specify and make public the minimum set of required processes, activities, and tasks, which constitute a supplier's conformance with the standard.

References

Works Cited

Arnold, S., and H. Lawson. 2004. "Viewing systems from a business management perspective." *Systems Engineering*, 7 (3): 229.

ISO. 2008. *Quality management systems -- Requirements.* Geneva, Switzerland: International Organisation for Standardisation. ISO 9001:2008.

ISO. 2004. *Environmental management systems -- Requirements with guidance for use.* Geneva, Switzerland: International Organisation for Standardisation. ISO 14001:2004

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes. Asian Pacific Council on Systems Engineering." Asia-Pacific Council on Systems Engineering (APCOSE) Conference, Keelung, Taiwan.

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

Primary References

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes." Proceedings of the 4th Asian Pacific Council on Systems Engineering (APCOSE) Conference, 4-6 October 2010, Keelung, Taiwan.

Additional References

None.

< Previous Article | Parent Article | Next Article (Part 4) >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.iso.org/directives

Part 4: Applications of Systems Engineering

Applications of Systems Engineering

Part 4 of the Guide to the SE Body of Knowledge (SEBoK) focuses on the application of systems engineering to the creation and life cycle management of various types of systems.

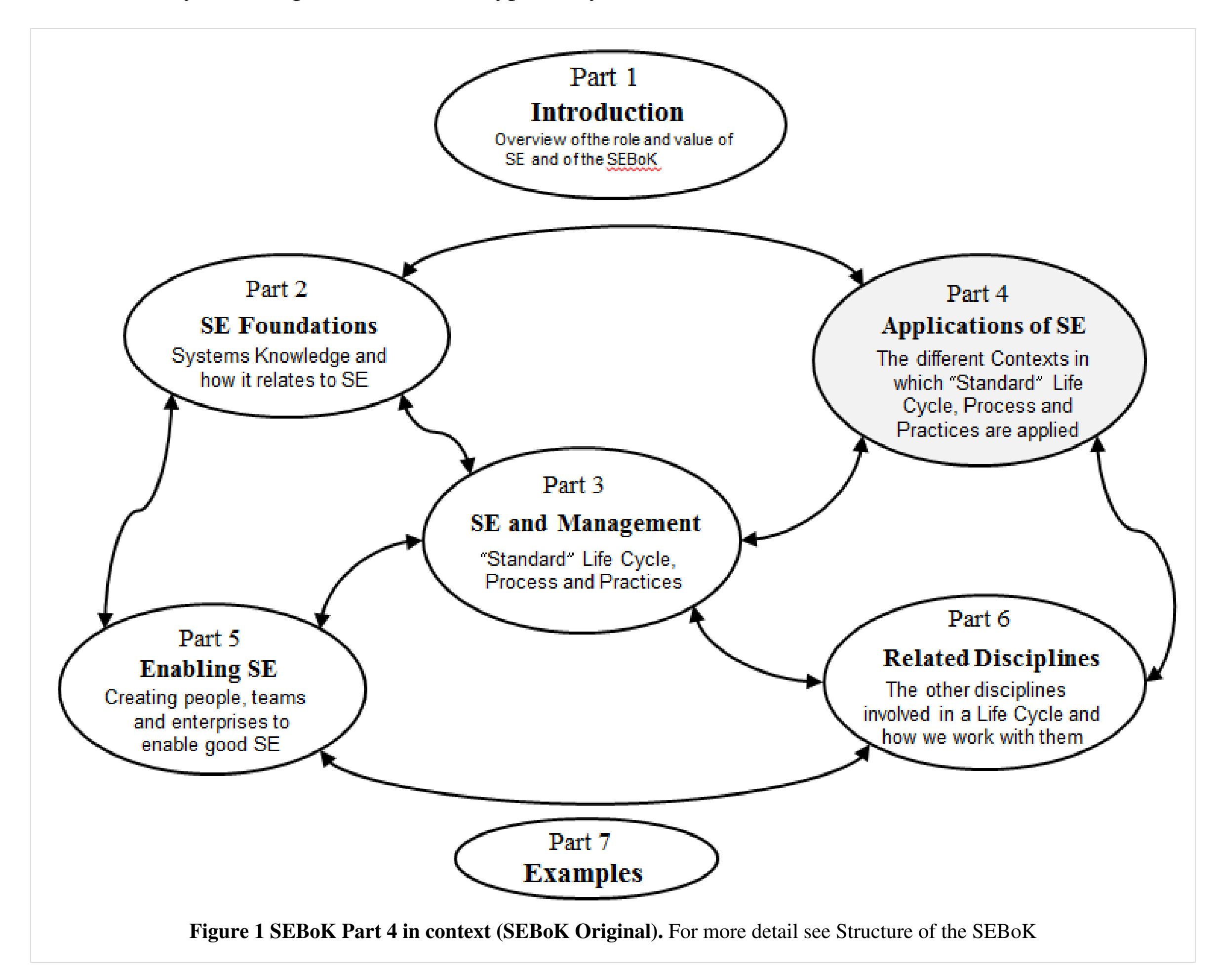


Figure 1 SEBoK Part 4 in context (SEBoK Original). For more detail see Structure of the SEBoK

In particular, the part covers product systems, service systems, enterprise systems, and system of systems (SoS). It also contains a knowledge area describing Healthcare SE as a domain extension of these general SE approaches. This is the first of a number of planned domain based extensions.

Knowledge Areas in Part 4

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 4 contains the following KAs:

- Product Systems Engineering
- Service Systems Engineering
- Enterprise Systems Engineering
- Systems of Systems (SoS)
- Healthcare Systems Engineering

Systems Engineering Application

The different ways in which each of these contexts shapes the application of the generic SE Life Cycle and Process knowledge in Part 3 is discussed in detail in the KA above.

It is important to note that none of the context above is intended to be wholly separate or mutually exclusive from the others. They should be seen as overlapping and related frameworks which provide a starting point for how generic SE might be used to fulfil real world needs. We might think of each as a model of how SE can work in the real world. Each gives advice on how to use generic SE life cycle and process knowledge against its own viewpoint. If necessary, each may also develop new or extended knowledge relevant to its context, which becomes part of the extended toolkit of SE. Like any set of models, each has its own simplifications, strengths and weaknesses. As a general principle we would always select the simplest model available which fits the purpose and use that. For a complex outcome the combination of a number of models may be needed.

The real world application of SE is no different. In most real projects combinations of Product, Service, Enterprise and SoS knowledge may be needed to achieve success. The extent to which these combinations are taken from pre-determined approaches vs the need for systems engineers to create such combinations are part of the application of SE is a key question for how SE is used. The final part of this knowledge, on how SE is applied in the real world, sits within the knowledge base of the various application domain. Some domains have a very detail set of procedures, guidelines and standards relevant to that domain, while others take general SE and apply it as needed using the judgement of those involved. In general, all domains have a little of both domain specific guidelines and experienced people. The SEBoK was originally written to be domain independent, other than through the application examples in part 7. To complete the SEBoK we intend to create a series of Domain Application KA. These will give an overview of how SE application maps to domain practice. They are aimed at both the general SE reader who wants to know more about a domain and those working within a domain.

The Healthcare SE KA contained in this version of the SEBoK is the first such domain specific extension of the SEBoK.

References

Works Cited

None

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Product Systems Engineering

Product Systems Engineering

Product systems engineering (PSE) is at the core of the new product development process (NPDP) that is needed to successfully develop and deploy products into different market segments. A market can be consumer based (e.g., private enterprises or general consumers) or it can be public (not-for-profit). Public markets address the strategic needs of a country or region, such as military, healthcare, educational, transportation, and energy needs. NPDP has two significantly overlapping and integrated activities:

1. **Systems engineering:** This includes concept generation, engineering design/development, and deployment
2. **Market development:** This includes market research, market analysis, product acceptance and market growth (diffusion), and rate of adoption

NPDP also includes manufacturability/producibility, logistics and distribution, product quality, product disposal, conformance to standards, stakeholder's value added, and meeting customer's expectations. The internal enterprise competence and capabilities such as customer support, sales & marketing, maintenance and repair, personnel training, etc., must also be taken into account.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Product Systems Engineering Background
- Product as a System Fundamentals
- Business Activities Related to Product Systems Engineering
- Product Systems Engineering Key Aspects
- Product Systems Engineering Special Activities

The Product Systems Engineering Background article discusses product types and the need for a product to be aligned with the business objectives of the enterprise. It also discusses the relationships between PSE and product development and technology development.

Various types of connections between product elements, and the concept of enabling systems are introduced in the Product as a System Fundamentals article. It also discusses product architecture, modeling, analysis, and integration with various specialty engineering areas.

Product launching and product offerings have close linkages to different business processes. The major linkages are to business development, marketing, product lines, quality management, project management, operations management, supply chain management, etc. These and other topics are described in the Business Activities Related to Product Systems Engineering article.

Products emerge when they are realized based upon a system definition. Realizing the system results in instances of the system that are either delivered as products to a specific acquirer (based upon an agreement) or are offered directly to buyers and users. Key Aspects of PSE are discussed in the Product Systems Engineering Key Aspects article which discusses aspects such as acquired vs. offered products, product lifecycle and adoption rates, integrated product teams (IPTs) and integrated product development teams (IPDTs), product architectures, requirements and

standards, etc.

The last article, Product Systems Engineering Special Activities, covers some of the special activities carried out by PSE during the different stages from concept through product deployment.

Key Terms and Concepts

Product

A product is an artifact that is created by some person or by some process such as a manufacturing process, software source code compilation and integration, building construction, creative writing process, or data processing.

In general, a business product is defined as a *thing produced by labor or effort*, or the *result of an act or a process*. It stems from the verb *produce*, from the Latin *prōdūce(re) (to) lead or bring forth*. Since 1575, the word *product* has referred to anything produced, and since 1695, the word *product* has referred to *a thing or things produced*.

In economics and commerce, products belong to a broader category of *goods*. The economic meaning of the word *product* was first used by political economist Adam Smith. In marketing, a product is anything that can be offered to a market that might satisfy a want or a need. In retail industries, products are called merchandise. In manufacturing, products are purchased as raw materials and sold as finished goods. Commodities are usually raw materials, such as metals and agricultural products, but a commodity can also be anything widely available in the open market. In project management, products are the formal definitions of the project deliverables that make up or contribute to delivering the objectives of the project. In insurance, the policies are considered products offered for sale by the insurance company that created the contract.

Product System

A product system is the combination of end products and the enabling products for those end products. This concept of a product system is illustrated in Figure 1. In the ANSI/EIA 632-2003 standard, just the term *system* is used, but the scope of the standard is clearly restricted to product systems.

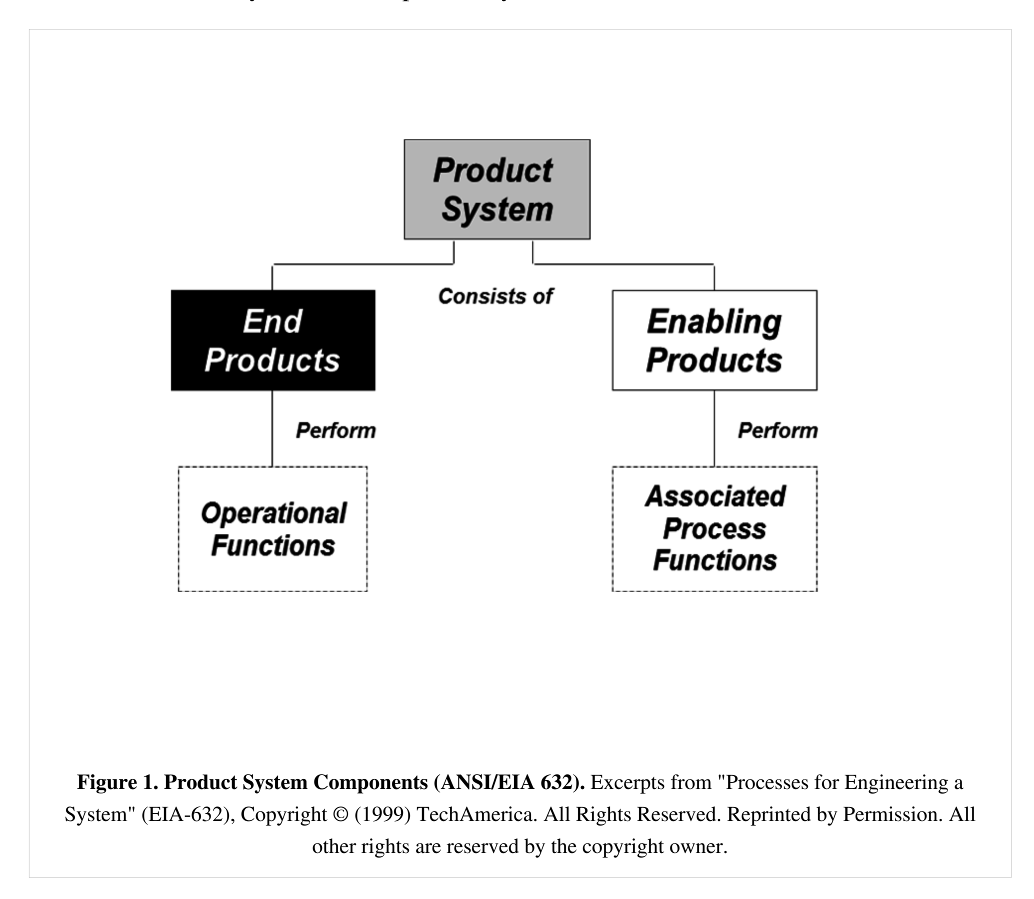


Figure 1. Product System Components (ANSI/EIA 632). Excerpts from "Processes for Engineering a System" (EIA-632), Copyright © (1999) TechAmerica. All Rights Reserved. Reprinted by Permission. All other rights are reserved by the copyright owner.

The end product can also be considered as a system with its own elements or subsystems, each of which has its own enabling products as illustrated in Figure 2. The product development process usually focuses only on the engineering of the end product. PSE is essential when the enabling products are by themselves complex or their relationship to the end product is complex. Otherwise, the use of a traditional product development process is sufficient.

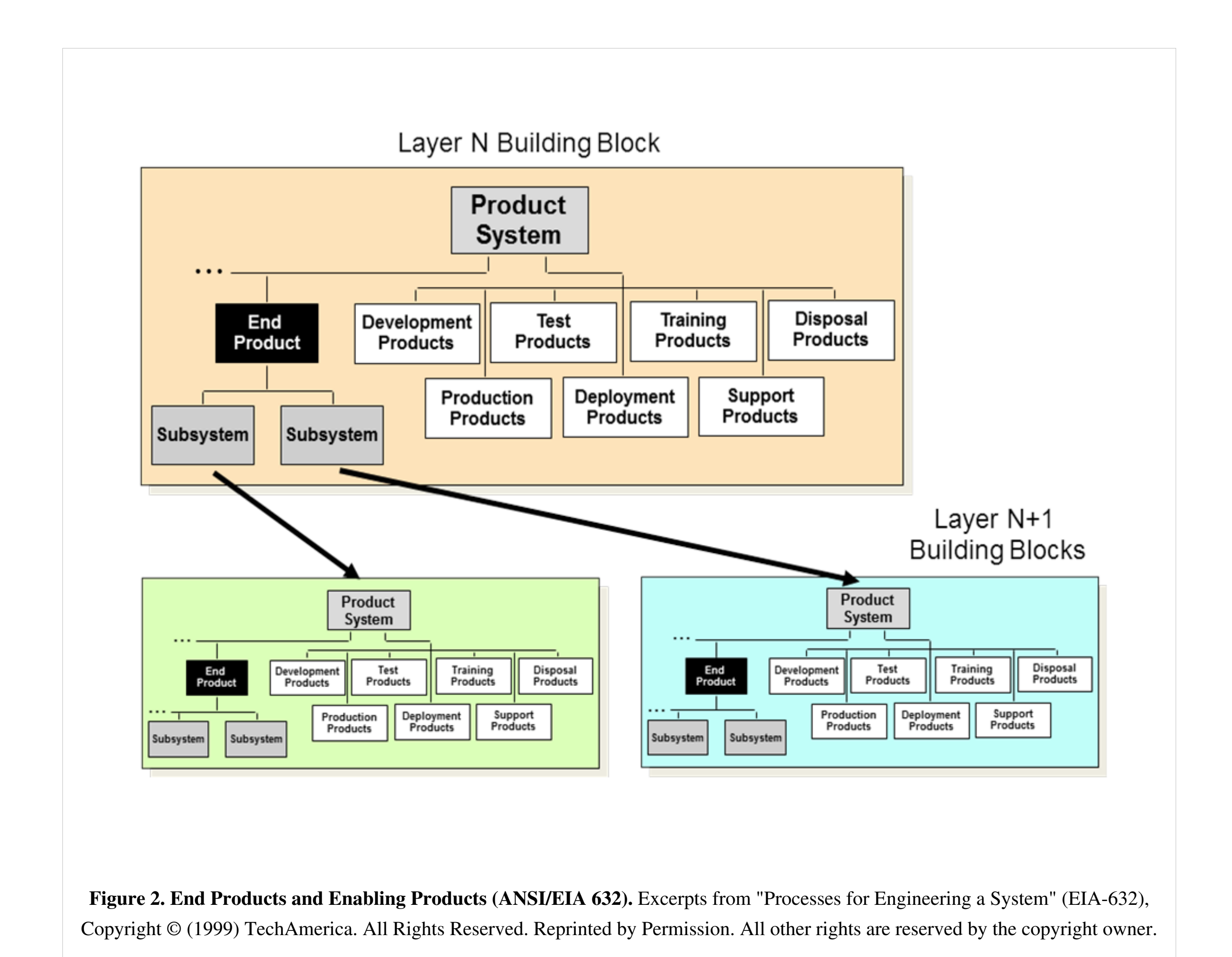


Figure 2. End Products and Enabling Products (ANSI/EIA 632). Excerpts from "Processes for Engineering a System" (EIA-632),

Product Realization System

There is a related system that enables the *realization* of the product system, which is the product realization system. It consists of *all the resources to be applied in causing the Intervention System [i.e., the product system, in this case] to be fully conceived, developed, produced, tested, and deployed* (Martin 2004). Lawson (2010) refers to this as a respondent system in the system coupling diagram. The intervention system is the system that is to be realized (or conceived and brought into being) in order to address some perceived problem in the context as shown in Figure 3.

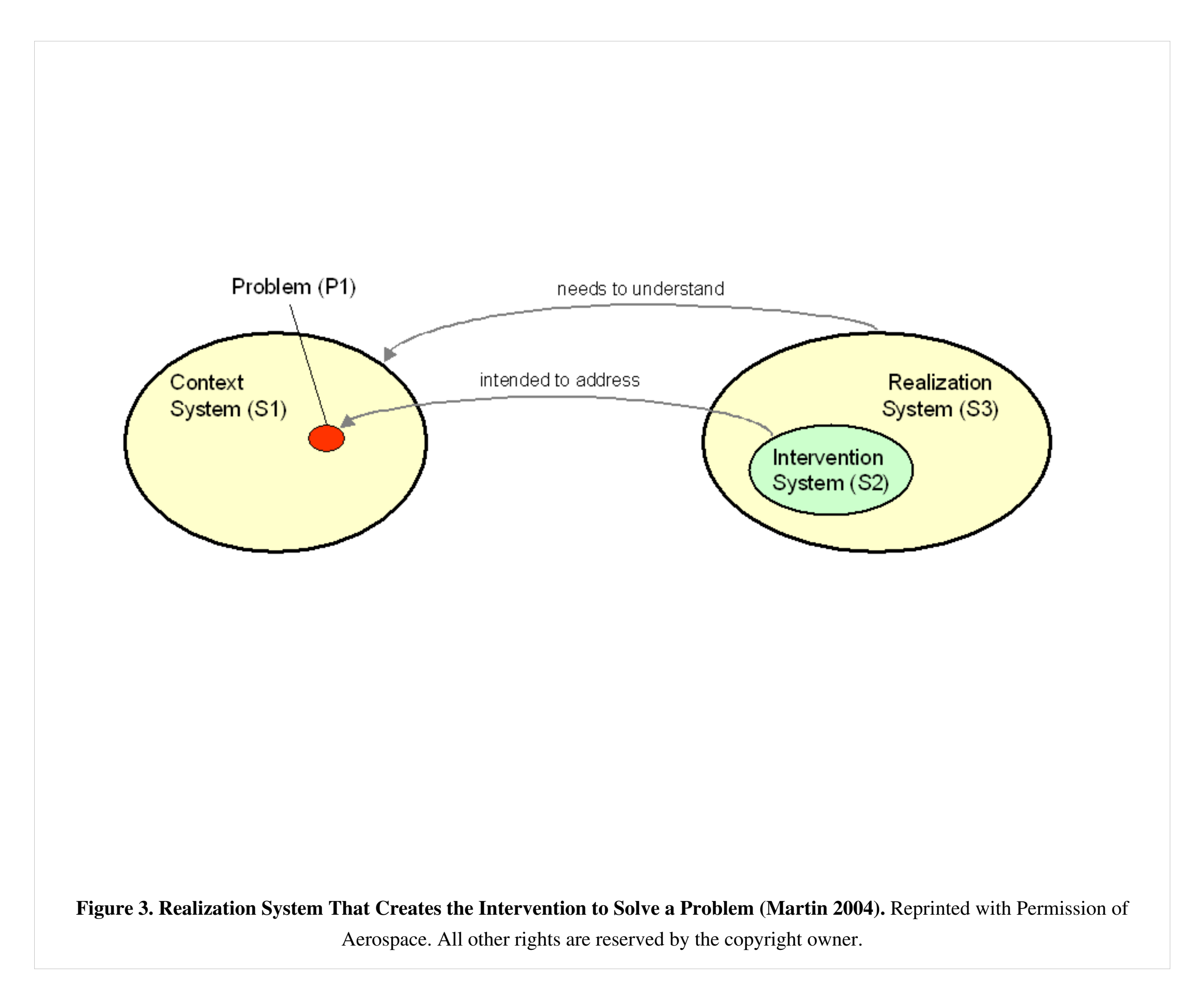


Figure 3. Realization System That Creates the Intervention to Solve a Problem (Martin 2004). Reprinted with Permission of Aerospace. All other rights are reserved by the copyright owner.

The realization system can be a service system (as described in knowledge area Service Systems Engineering) or an enterprise system (as described in the knowledge are Enterprise Systems Engineering). When the realization system is a service system, then the service could partially realize the system by just designing the product system without developing or creating it. This design service system can pass the design to a manufacturing service system that turns the design into a physical artifact. Usually an enterprise is established to orchestrate the disparate services into a cohesive whole that is efficiently and effectively performed to achieve the strategic goals of that enterprise.

The product realization system utilizes a product realization process as described in (Magrab et al 2010) or a product development process as described in (Wheelwright and Clark 1992).

Product Sustainment System

When the realization system delivers the product system into its intended environment, the product often needs a set of services to keep that product operational. This other system, when needed, is called the product sustainment system. It consists of various enabling products and operational services. The sustainment system in relation to the realization system and the deployed product system is illustrated in Figure 4. Notice that the realization may need to develop or modify the sustainment for the particular intervention (product) system under development.

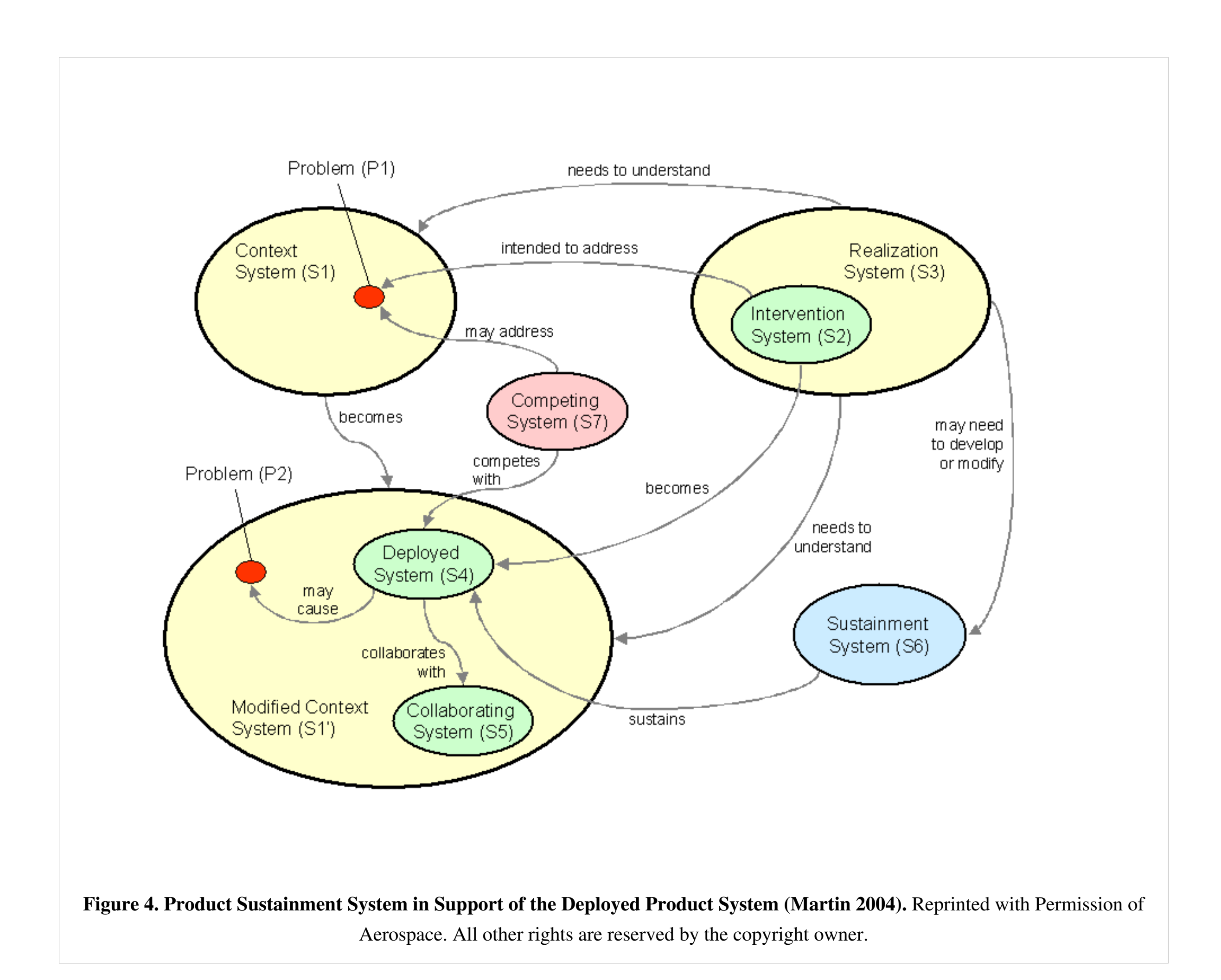


Figure 4. Product Sustainment System in Support of the Deployed Product System (Martin 2004). Reprinted with Permission of Aerospace. All other rights are reserved by the copyright owner.

Product Systems Engineering, Service Systems Engineering and Enterprise Systems Engineering

PSE is in line with Traditional Systems Engineering (TSE) as captured in most textbooks on the subject, such as Wasson (2006), Sage and Rouse (2009), and Blanchard and Fabrycky (2011). However, they do not cover the full breadth of PSE since they tend to focus on hardware and software products only. Other kinds of products to be engineered include personnel, facilities, data, materials, processes, techniques, procedures, and media (Martin 1997; Lawson 2010). Further discussions on the distinctions between the various kinds of products is provided in the Product Systems Engineering Background article. Product system domains could be data-intensive (e.g. transportation system), facilities-intensive (e.g. chemical processing plant), hardware-intensive (e.g. defense systems), or technique-intensive (e.g. search and rescue system). Most product systems are a composite of several different kinds of products that must be fully integrated to realize the complete value added potential for the different stakeholders.

When compared to Service Systems Engineering (SSE) and Enterprise Systems Engineering (ESE), PSE has some unique considerations:

- Often a product is part of a product line where both the product line and the products that make up that product line must be engineered simultaneously.
- Products are often composed of parts and sub-assemblies produced by several suppliers. This entails the need to work closely with the supply chain to ensure a successful product offering.
- Large complex products often require a lengthy and complicated series of steps for assembly, integration and test. During integration, many of the key assumptions made during the initial product design could be challenged.

- Products will usually require certification as to their safety or other factors like energy conservation and environmental compatibility. Electronic products often require certification to ensure electromagnetic compatibility and limited electronic emissions into the radio frequency spectrum. Transportation products require certification for safe operations.
- Products often have a complicated distribution network since they are not always developed where the end user may require it. There could be depots, warehouses, multi-modal transportation, wholesalers, dealers, and retail stores as part of this distribution network. This introduces challenges in delivery, maintenance and support of the product.
- Products must be engineered along with the realization system and the sustainment system. Sometimes it is necessary to make tradeoffs between the features and functions of the product system, the realization system and the sustainment system.

These considerations and others will be addressed in the articles under this knowledge area. One of the responsibilities of ESE is to manage these various considerations across multiple product lines across the enterprise while maximizing profits and customer satisfaction. SSE is often dependent on the products resulting from the PSE. A service will often be based on a product infrastructure that includes elements like data processing, hardware, software, data storage, data entry devices, display devices, service delivery facilities and techniques, service desk technicians, maintenance personnel, service offering catalogs and printed materials. Each of these products in the service infrastructure may need to be separately engineered using its own PSE lifecycle.

Creating Value

An enterprise that creates products must also create value in the eyes of the customer; a value that exceeds the cost of that product. This applies to both private and public enterprises operated for profit or not-for-profit. The creation and delivery of products may be the result of an acquisition agreement or an offering directly to buyers or users. To remain competitive, enterprises also need to understand the effects of the global economy, trends in industry, technology development needs, effects of new technology breakthroughs, market segments creation and their expectations, and most importantly, ever evolving customer expectations.

Ring (1998) defines a system value cycle with three levels that a systems approach must consider to deliver real world benefit:

1. stakeholder value
2. customer need
3. system solution

Value will be fully realized only when it is considered within the context of time, cost, and other resources appropriate to key stakeholders.

Aligning product characteristics with associated operational activities

The user of a product views the product as an asset that can be utilized in one's own systems of interest (Lawson 2010). Thus, in supplying the product, the expected form of operation becomes a driving factor in determining the characteristics of the product. In several contexts, in particular for military related products, the desired operational activities are termed concept of operations (ConOps) and in the case of commercial enterprises the intended use of the system is described through some form of *Market Service Description* of the product. The intended use of the product is market/customer driven and so the product characteristics must be aligned with the operational intent.

Architectures as basis for value assessment

Architectures can be used by enterprises to shift product development from individual products to an underlying product line architecture that incorporates the flexibility required by the enterprise to rapidly tailor new technologies and features to specific customer requirements (Phillips 2001). In determining the architecture of the product system, various alternative designs may arise. Each of the architecture alternatives is to be evaluated with respect to its value contribution to end users and other stakeholders.

Role of evaluation criteria in selection between product alternatives

In assessing the product system value, one must consider the measures that are to be used to determine the *goodness* of the product alternatives (alternative architectures and technologies) with respect to producibility, quality, efficiency, performance, cost, schedule and most importantly the coverage provided in meeting the customer's requirement or market opportunity.

Role of tradeoffs in maximizing value

The evaluation of alternatives must include the tradeoffs between conflicting properties. For example, in striving for superior quality and efficiency, tradeoffs must be made with respect to schedule and cost. See article on Measurement in Part 3. Tradeoffs are made during different stages of the development process: at the product or system level, at the subsystem and architecture definition level, and at the technology level (Blanchard and Fabrycky 2011).

There are a variety of methods for performing tradeoff analysis such as: utility theory, analytic hierarchical process, the Pugh selection method, multi-objective decision, multi-attribute utility analysis, and multi-variate analysis. For software, the Software Engineering Institute (SEI) provides 'The Architecture Tradeoff Analysis Method (ATAM)' (Kazman et al., 2000) for evaluating software architectures relative to quality attribute goals. ATAM evaluations expose architectural risks that potentially inhibit the achievement of an organization's business goals. The ATAM not only reveals how well an architecture satisfies particular quality goals, but also provides insight into how those quality goals interact with each other and how they trade off against each other.

Expanding role of software in creation of product value

Software has an increasing role in providing the desired functionality in many products. The embedding of software in many types of products (such as transportation vehicles, home appliance, and production equipment) accounts for an ever increasing portion of the product functionality. The current trend is the development of a network of systems that incorporate sensing and activating functions. The use of various software products in providing service is described in the Service Systems Engineering knowledge area.

References

Works Cited

ANSI/EIA. 2003. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA). ANSI/EIA 632-2003.

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Kazman, R., M. Klein, and P. Clements. 2000. *ATAM: Method for Architecture Evaluation*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU). CMU/SEI-2000-TR-004, ESC-TR-2000-004.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications.

Magrab, E., S. Gupta, P. McCluskey, and P. Sandborn. 2010. *Integrated Product and Process Design and Development - The Product Realization Process*. Boca Raton, FL, USA: CRC Press.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products*, 1st ed. Boca Raton, FL, USA: CRC Press.

Martin, J. 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, 2004, Toulouse, France.

Phillips, F. 2001. *Market-oriented Technology Management - Innovating for Profit in Entrepreneurial Times*. Berlin, Germany: Springer-Verlag.

Pugh, S. 1990. *Total Design: Integrated Methods for Successful Product Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Ring, J. 1998. "A Value Seeking Approach to the Engineering of Systems." Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, 1998, San Diego, CA, USA. p. 2704-2708.

Sage, A., and W. Rouse. (eds.) 1999. *Handbook of Systems Engineering and Management*. Hoboken, NJ, USA: John Wiley and Sons, Inc.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. New York, NY, USA: John Wiley & Sons.

Wheelwright, S., and K. Clark. 1992. *Managing New Product and Process Development: Text and Cases*. Columbus, OH, USA: Free Press.

Primary References

ANSI/EIA. 2003. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA). ANSI/EIA 632-2003.

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

Magrab, E., S. Gupta, P. McCluskey, and P. Sandborn. 2010. *Integrated Product and Process Design and Development - The Product Realization Process*. Boca Raton, FL, USA: CRC Press.

Martin, J. 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, 2004, Toulouse, France.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. New York, NY, USA: John Wiley & Sons.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Product Systems Engineering Background

Product Types

A system is by definition composed of elements that interact. The system itself usually is an element of a larger system, and you can often also view each element as a system on its own.

A system element consists of one or more products. Products need to be produced or acquired. Some can be acquired or procured as-is, without need for fabrication or modification. Others need to be engineered, and in some cases systems-engineered (Martin 1997). Basic product types are depicted in the figure below.

Types of products are not limited to hardware or software. Many other types of products perform functions necessary to meet stakeholder needs. Some are only relevant to certain industries or domains, such as structures for civil engineering, or ships for shipping or the naval domain. Systems engineers must remember not to allocate the required behavior for a system to hardware and software elements alone.

While we may associate the idea of a product with concrete objects like computer chip, phones, aircraft, or even command, control and communications centers, an organization or a process can also be a product. Sometimes a product is not complex enough to justify performing Product SE, and only needs product design engineering. Enterprise SE and Service SE should determine whether a product needs Product SE.

Product Taxonomy

For any system being developed, the systems engineers must decide what are the right elements to be included. This is not self-evident, because basic product types are not necessarily mutually exclusive. For example, some would consider that facilities contain hardware and people. Others would consider facilities to be separate from hardware and people. Some would include material as part of hardware, while others would not. Creating a taxonomy of product types can help the systems engineer think clearly and thoroughly about what components to include.

Business Objectives and Products

When it develops and launches a new product, an enterprise must align that product with its business goals, internal capabilities, and competition. It must align the end product with the systems expected to realize and sustain it.

The new product concept must be based on analysis that, besides product potential, also explores the ability of the enterprise to exploit that potential, including factors like organizational culture, focus, goals, and processes. Present and future markets and technology must be analyzed. So must several dimensions of competition: competitors' offerings and their plans, for entry into new markets and for product expansion including new functionality, features, or services. These, and the ability of the enterprise to react to them, must also be monitored for the enterprise to remain competitive in the long term.

Accelerating economic globalization since the 1970s has forced enterprises to respond to global needs, not just local or regional ones. Enterprises in the resulting hyper-competitive business environment must analyze their financial goals, their market positions, and the business segments in which they participate, in order to understand what products are required.

This is true for completely new products, and also for product enhancements, penetration of new markets, and growth within existing markets.

Relationship between Product Systems Engineering and Product Development

Product development is the process of bringing a new product to market. Product SE (PSE) considers the complete product system–that is the product in the context of all its enabling elements. PSE takes a full life cycle perspective, "from cradle to grave" or "dust to dust."

Technology-based product development may be thought of as coming from two sources. One, where innovation enhances existing technology, is aimed at relatively short-term market windows. The other involves long-term research to identify the technology developments required to realize the concept. These may be technologies whose availability is not foreseen in the near future, meaning that substantial investment and long lead times may be required before the proof of concept, initial operational capabilities (IOCs) or prototyping stages are reached, let alone the commitment to realize the actual product offering. Some authors claim that the systems engineering process and the new product development (NPD) process for this second source are one and the same.

It is from the second source that strategic initiatives (long-term applied research) realize new products in areas like military aircraft or bioengineering. When research resolves fundamental questions on matters of science or national/regional interest technology breakthroughs occur.

This article concentrates on the first source of technology-based product development, that is, the one driven by ever-evolving market needs to enhance existing technology.

Product Development Patterns

When existing or near-future technology innovations are exploited to generate new product ideas, product development may follow any one of following scenarios (Phillips 2001):

- Product development may use well-established technologies to help the enterprise improve the efficiency of current operations.
- Product development may use well-established technologies to help the enterprise into new kind of operations.
- Product development may use leading edge technologies to improve the efficiency of current operations.
- Product development may use leading edge technologies to help the enterprise into new kinds of operations.

The product itself may simply be a modification of an existing product or its presentation, it may possess new or different characteristics that offer additional benefits to the customer, and/or it may be entirely new and satisfy a newly-defined customer want or market niche (http:/ / www. businessdictionary. com/ definition/ product-development.html [1]).

Existing realization or sustainment systems may not be adequate to develop a given product. For example, it might be necessary to change development practices, use different testing methods or facilities, or upgrade manufacturing equipment and procedures. There might need to be improved customer support procedures and newly trained support personnel, upgraded maintenance facilities and tools, or modified spare parts delivery techniques.

Market Pressures

The product development process must be dynamic, adaptive, and competitive on cost, time-to-market, performance, and quality on a global scale. This is because in the global economy continuous technology innovation and constantly evolving markets and customer needs demand a rapid response.

Products themselves are often multidisciplinary in nature; product development must have close participation, not only from the different specialty engineering fields (mechanical, electrical, industrial, materials, and so on), but also from the finance field to analyze the total cost of development (including production), marketing and sales to understand market behavior and acceptance, manufacturers and distributors, and legal and public relations experts.

All this has mandated enterprises to assess how they create their products and services. The result has been an effort to streamline the development process. One example of this is seen by the deployment of integrated product teams (IPTs) sometimes known as integrated product development teams (IPDTs).

Product Systems Engineering

Product systems engineering strives for the efficient use of company resources in order to achieve business objectives and deliver a quality product. Product systems engineering activities range from concept to analysis to design and determine how conceptual and physical factors can be altered to manufacture the most cost-effective, environmentally friendly product that satisfies customer requirements. Engineering the product system requires an interdisciplinary approach that includes an analysis of the product and its related elements such as manufacturing, maintenance, support, logistics, phase-out, and disposal; these are all activities which belong to either the realization system or the sustainment system. The proper application of systems engineering and analysis ensures the timely and balanced use of human, financial, technological assets, and technology investments to minimize problems, harmonize overall results, and maximize customer satisfaction and company profits.

Products are as diverse as the customers that acquire them and there are no universally accepted methods, processes, and technologies (MPTs) for end-to-end analysis of products and their supporting subsystems. Every product needs to adapt existing MPTs based on prior experiences and best practices, such as Toyota (Hitchens 2007), MITRE (Trudeau 2010), and NASA (NASA SELDP 2011). Product systems engineering helps develop the end-to-end analysis of products and sub-systems by performing the following tasks:

- determining the overall scope of needs for the product system;
- defining product and system requirements;
- considering all interactions between the different elements of the product system;
- organizing and integrating engineering disciplines; and
- establishing a disciplined approach that includes review, evaluation and feedback, and ensures orderly and efficient progress.

Constantly evolving needs and requirements, along with constant technology innovations, may render a committed product development obsolete even before deployment. This has led to debate among systems engineering professionals on the need for the systems engineering process to become more rapidly adaptable. Platform-based solutions to resolve some of these challenges (infrastructure as a service, platform as a service, and software as a service) are being studied and proposed (MITRE 2010; Boehm 2010).

Integrated Product Development Process

The integrated product development process (IPDP) starts with understanding market needs and developing a strategy that creates products that satisfy or exceed customer expectations, respond to evolving customer demands, adapt to changing business environments, and incorporate systems thinking to generate novel ideas and co-create value with extensive stakeholders' participation. IPDP is a continuously evolving process that strives to realize products whose cost, performance, features, and time-to-market help increase company profitability and market share. Magrab, et al. (2010) discussed the IPDP in terms of four different stages; Figure 1 provides a snapshot of an IPDP and the main tasks carried out at each stage.

Stage I: Product Identification

During the product identification stage, the enterprise aims to identify an enterprise-wide strategy that flows down to individual product strategies resulting in a good business investment for the company. During this stage addressable markets for the product are identified in addition to geographical coverage of the product. The developments through this stage result in demonstration of strong customer need, determination of potential markets and geographic scope, the fitness of enterprise core capabilities to the product strategy, business profitability (return on investment, profit & loss), etc.

During this stage an integrated product team (IPT) first develops the IPDP for the project, usually by tailoring a corporate IPDP standard. The IPT assesses required technology innovation, feasibility of existing technologies, estimated time and cost of technology development, and the risks associated with markets, finances and technologies

risk, etc. This stage also takes into account inputs from the continuous improvement (CI) process to develop new features, enhancements in existing products to address new market needs or customer demands.

Stage II: Concept Development

The main goal of the Concept Development stage is to generate feasible concepts designs for the potential product and develop MPTs that will satisfy the product's performance goals of economic viability and customer satisfaction. These concept designs must ensure that the company's core competencies can satisfy the requirements to produce the products while keeping into account the market viability, manufacturability, and technical feasibility through an extensive analysis of alternative process.

During this stage SE supports the IPT in identifying different operational scenarios and modes of operation, functional requirements of the products, technology risks and performance risks, and the main components of the products and required interfaces among them, etc. This stage involves a highly interactive and iterative exchange of concepts among several IPTs and depending on complexity of the products, a Systems Engineering Integration Team may be required to ensure analysis of all the possible solutions. During this stage inputs from the CI process helps analyze new technologies/processes including upgrades to existing technologies, and create products that results in enhanced customer experiences.

Stage III: Design and Manufacturing

During the design and manufacturing stage the actual product is realized and manufactured. This stage starts with creating engineering drawings for the product, product configuration items specs, "design for X" (DFX), manufacturing design plans, production plans and schedules, test production run to ensure that product meets customer requirements and quality criteria, and a plan for full production, logistics and distribution.

During this stage the product design & manufacturing engineering team works closely with operations managers to create MPTs to manage the technical effort for the product from an end-to-end perspective. Some of the SE activities during this stage include product integration, verification and validation plans; modeling, simulation, test & evaluation of the product system under critical scenarios; launch readiness plans including end-user test plans, operational readiness, etc. During this stage MPTs are developed and documented for proper handling of defective parts, processes, or functionalities. The CI process inputs include product and process performance enhancements and sustained life-cycle operations support.

Stage IV: Product Launch

During the product launch stage the product is delivered to its potential markets. During production and deployment, MPTs are developed to ensure that the product meets its quality goals, satisfies customer requirements, and realizes the business plan goals. This requires provisions for customer care, logistics, maintenance, training etc., and a CI process to monitor product and product system technical performance and product quality. The CI process is realized through extensive data collection using customer satisfaction surveys and remotely or manually observing, recording, and analyzing process performance metrics, technical performance measure, quality metrics, etc.

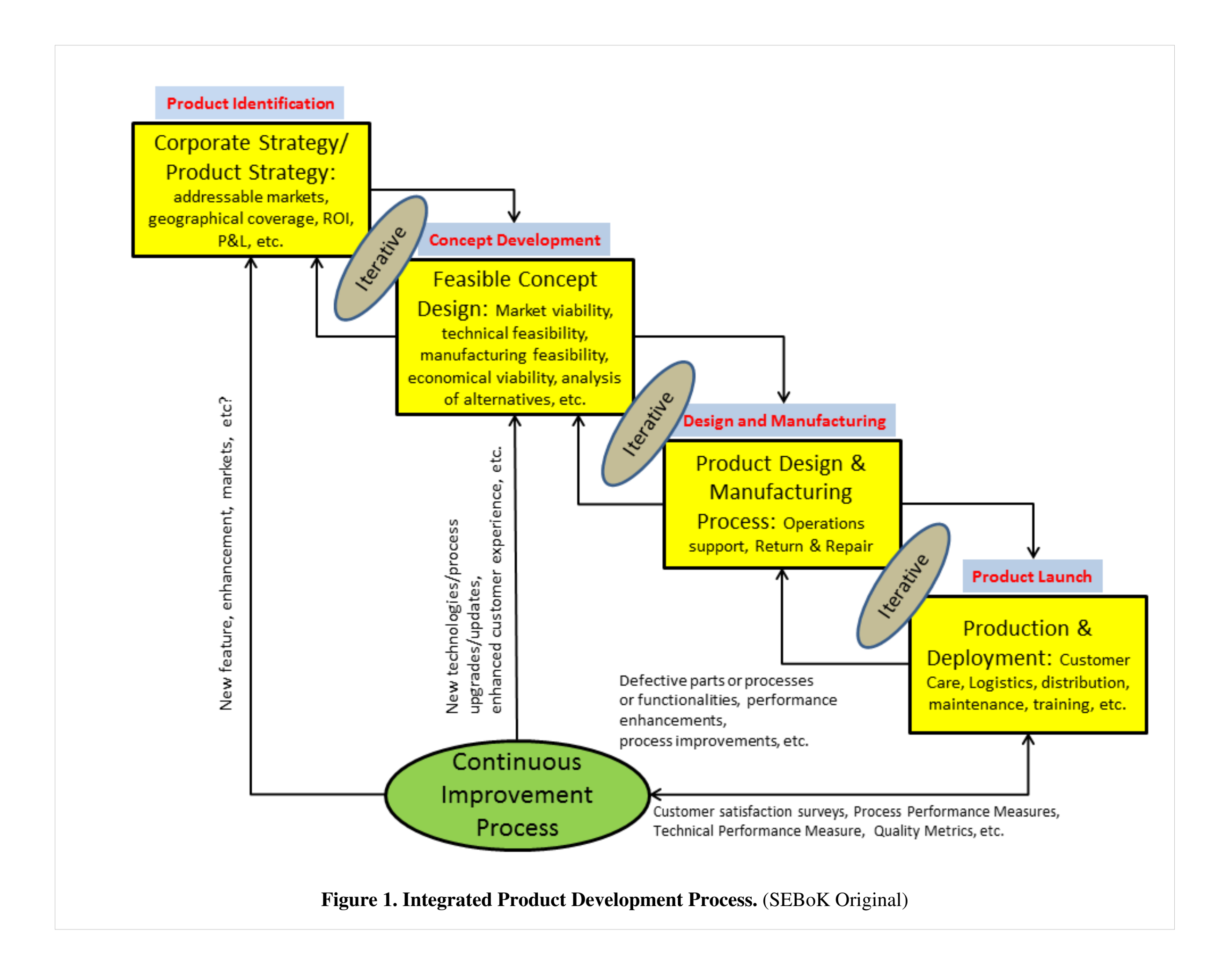


Figure 1. Integrated Product Development Process. (SEBoK Original)

Relationship between Product Systems Engineering and Technology Development

As technological advancement accelerates, product life cycles become shorter, especially for high technology products. As a result, enterprises risk having outdated or obsolete products that have lost pace with markets trends, technology trends, or customer expectations.

Product systems engineering should bring awareness of technology changes and trends to the analysis of new product ideas or innovations. This affects the time and cost inputs into the technical feasibility analysis of the product. The result should include a road map of required technology developments, which is then used to create the overall road map for the new product offering.

In these cases, new product ideas impose requirements on new technology developments.

On the other hand, when technology developments or breakthroughs drive product innovation or the generation of new markets, the technology developments may also generate requirements on product features and functionalities. Factors which dictate decisions about introducing products include the technology readiness levels (TRL), the integration readiness levels (IRL), the manufacturing readiness levels (MRL), the system readiness levels (SRL), and the operational readiness of the enterprise to launch the product system. See the "Readiness Levels" section in the Product Systems Engineering Special Activities article.

Understanding the entities (i.e., components or elements) that compose the product is not a trivial task for systems engineers. It is not unusual for a new product to require developments in several technologies, including new materials, electronic components, software, maintenance and repair procedures, processes, or organizational structures. All of these developments must be factored into the IPDP for the successful deployment and proper use of the product.

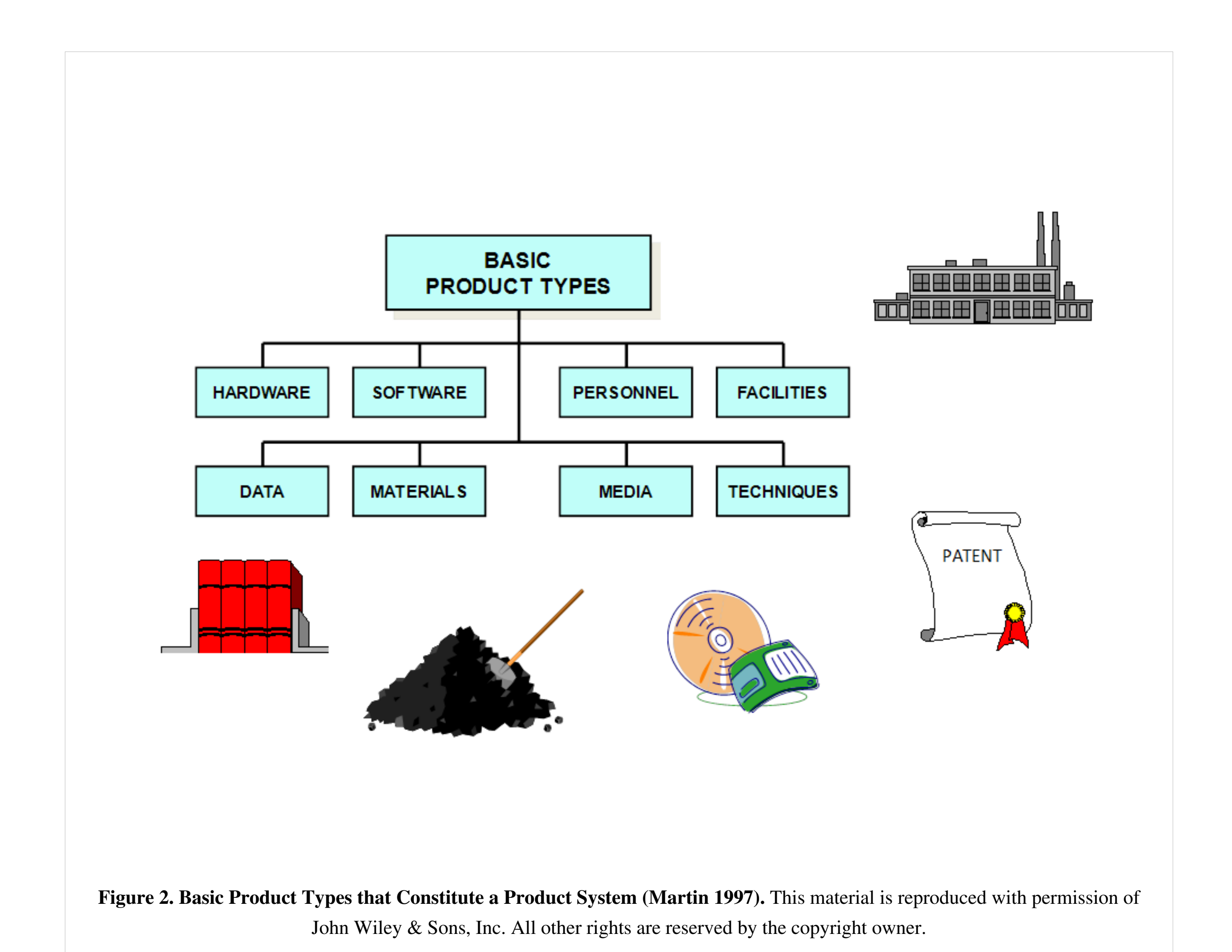


Figure 2. Basic Product Types that Constitute a Product System (Martin 1997). This material is reproduced with permission of John Wiley & Sons, Inc. All other rights are reserved by the copyright owner.

Product Type Examples

Examples of each product type are shown below (Martin 1997).

Table 1. Product Types (Martin 1997). This material is reproduced with permission of John Wiley & Sons, Inc.

Type	Examples
Hardware	Computer processor unit, radar transmitter, satellite payload, telephone, diesel engine, data storage device, network router, airplane landing gear
Software	Computer operating system, firmware, satellite control algorithm, robot control code, telephone switching software, database application
Personnel	Astronaut, computer operator, clerk, business executive, Laika (the cosmonaut dog), bus driver, cashier, maintenance technician
Facilities	Space rocket launch pad, warehouse building, shipping docks, airport runway, railroad tracks, conference room, traffic tunnel, bridge, local area network cables
Data	Personnel records, satellite telemetry data, command and control instructions, customer satisfaction scores
Materials	Graphite composite, paper, gold, concrete, stone, fiberglass, radar absorption material, cladded metals, integrated circuit substrate, magnetic memory core
Media	Data storage media (tape, disc, memory card), signal transport media (twisted pair wire, fiber optic cable, RF spectrum), communications media (television, radio, magazines), social media (blogs, Twitter, Facebook)
Techniques	Soldering, trouble trick response process, change notice handling, telephone answering protocol, project scheduling, data sorting algorithm

Materials could be thought of as basic raw materials, like steel, or as complex materials, like cladded metals, graphite composites, or building aggregate material. Personnel are not normally thought of as a "product," but that can change depending on the type of system in question. The National Aeronautics and Space Administration (NASA) space program "system" certainly produces astronauts. When personnel are considered system(s), it is not usually possible to simply find and hire personnel with the requisite knowledge, skills, and experience. These personnel "products" can often be developed using a product SE approach (Martin 1996). For example, you could specify requirements (i.e., required knowledge, skills, and experience) for each person that is part of the system. Interfaces can be specified for each person, and an assessment can be made as to the maturity of each person (i.e., each potential product). These are a few examples of how product SE can be applied to personnel products.

In enterprise systems engineering, we may need education and training systems to make up a part of our personnel system in order to produce people with the right competencies and capabilities.

References

Works Cited

Academy of Program/Project and Engineering Leadership (APPEL). 2009. *NASA's Systems Engineering Competencies*. Washington, DC: US National Aeronautics and Space Association. Available at: http://www.nasa.gov/offices/oce/appel/pm-development/pm_se_competency_framework.html [2].

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

Boehm, B. 2010. *Systems 2020 Strategic Initiative*. Hoboken, NJ, USA: Systems Engineering Research Center (SERC), SERC-2010-TR-009.

Grady, J. 2010. *Systems Synthesis - Product and Process Design*. Boca Raton, FL, USA: CRC Press.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Magrab, E., S. Gupta, P. McCluskey, and P. Sandborn. 2010. *Integrated Product and Process Design and Development - The Product Realization Process*. Boca Raton, FL, USA: CRC Press.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products*, 1st ed. Boca Raton, FL, USA: CRC Press.

MITRE. 2010. *Platform as a Service: A 2010 Marketplace Analysis*, Cloud Computing Series. Bedford, MA, USA: Systems Engineering at MITRE.

Morse, L., and D. Babcock. 2007. *Managing Engineering and Technology*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Phillips, F. 2001. *Market Oriented Technology Management: Innovating for Profit in Entrepreneurial Times*. New York, NY, USA: Springer.

Trudeau, P.N. 2010. *Designing and Enhancing a Systems Engineering Training and Development Program*. Bedford, MA, USA: The MITRE Corporation.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. New York, NY, USA: John Wiley & Sons.

Primary References

Grady, J. 2010. *Systems Synthesis - Product and Process Design*. Boca Raton, FL, USA: CRC Press.

Magrab, E., S. Gupta, P. McCluskey, and P. Sandborn. 2010. *Integrated Product and Process Design and Development - The Product Realization Process*. Boca Raton, FL, USA: CRC Press.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and product*, 1st ed. Boca Raton, FL, USA: CRC Press.

Additional References

Academy of Program/Project and Engineering Leadership (APPEL). 2009. *NASA's Systems Engineering Competencies*. Washington, DC: US National Aeronautics and Space Association. Available at: http://www.nasa.gov/offices/oce/appel/pm-development/pm_se_competency_framework.html [2].

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

Boehm, B. 2010. *Systems 2020 Strategic Initiative*. Hoboken, NJ, USA: Systems Engineering Research Center (SERC), SERC-2010-TR-009.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

MITRE. 2010. *Platform as a Service: A 2010 Marketplace Analysis*, Cloud Computing Series. Bedford, MA, USA: Systems Engineering at MITRE.

Morse, L., and D. Babcock. 2007. *Managing Engineering and Technology*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Phillips, F. 2001. *Market Oriented Technology Management: Innovating for Profit in Entrepreneurial Times*. New York, NY, USA: Springer.

Trudeau, P.N. 2010. *Designing and Enhancing a Systems Engineering Training and Development Program*. Bedford, MA, USA: The MITRE Corporation.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. New York, NY, USA: John Wiley & Sons.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.businessdictionary.com/definition/product-development.html

[2] http://www.nasa.gov/offices/oce/appel/pm-development/pm_se_competency_framework.html

Product as a System Fundamentals

This article introduces fundamental concepts of product systems.

Product Elements and Connections

Product systems consist of product elements and two kinds of connections: connections among elements, and connections between elements and things in the system environment. That portion of the environment that can be influenced by the system or that can influence the system is called the “context.”

Connections between elements contain interactions and relationships (Hybertson 2009). A connection is more than a mere interface.

Interactions occur across *interfaces* between the elements inside or outside the system, and can be defined as exchanges of data, materials, forces, or energy. Connections with an interactive nature can be represented in various engineering artifacts: schematic block diagrams, data flow diagrams, free body diagrams, interface control diagrams, port specifications, energy transfer diagrams, and so on. Product systems engineering (PSE) usually defines interactions in an interface control document, interface design document, interface requirements document, or the equivalent.

Connections also encompass relationships between elements. These relationships can be spatial, motion-related, temporal, or social.

Spatial relationships:

- one element is underneath another
- two elements are *x units* apart
- one element is inside another

Motion-related relationships:

- the relative velocity of two elements is *v units*
- the relative acceleration between two elements is *a units*

Temporal relationships:

- one element exists before another
- two elements must exist at the same time
- two elements must be separated in time by *t units*

Social relationships:

- a human element feels a particular way about a system
- a human element owns another (non-human) element
- a human element understands the operation of a system in a particular way

Relationships that are not about time can still change over time. For example, an element that is inside another element during one mode of operation can be outside of it during a different mode of operation. Therefore, one should not assume that non-temporal relationships are necessarily static in time.

Relationships can be represented in engineering artifacts, including the timing diagram, timeline diagram, mission reference profile, capability road map, and project schedule chart.

Social relationships include the implicit or explicit social obligations or expectations between the roles that human elements play in a system. These roles may be assigned different authorities, responsibilities, and accountabilities. See the discussion on organization behavior in the article Team Dynamics. Organizational behavior theories and human factors may need to be considered when engineering such a product system.

There can also be social relationships between the humans and the non-human elements of the system. This may involve how the human “feels” about things in the system or perhaps even the system as a whole. Humans inside or outside the system-of-interest may have different degrees of “understanding” with respect to how the system operates, its limitations and capabilities, and the best way to operate it safely and effectively. The “ownership” relationship can be important in determining things like who can operate or change some configuration or mode of the system.

There are many such social relationships in a product system that are often ignored or misunderstood when performing PSE. Social relationships can affect the overall performance or behavior of a product system to the point of determining its success or failure.

Core Product and its Enabling Products & Operational Services

A variety of systems (themselves being products or services) enable the development, delivery, operation and eventual disposal of a product, as shown in Figure 1. The concept of enabling systems is defined in the ISO/IEC 15288 standard (2015).

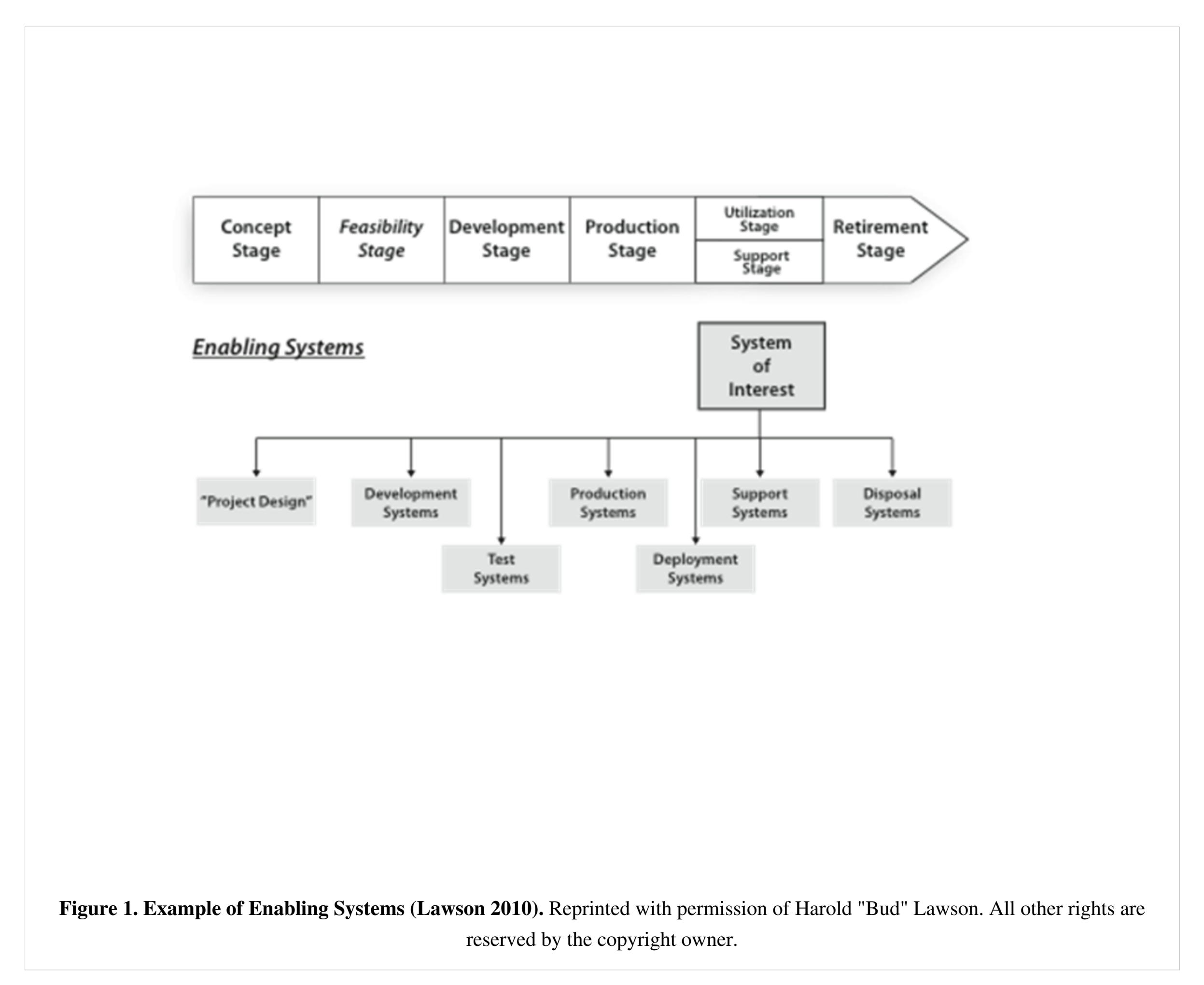


Figure 1. Example of Enabling Systems (Lawson 2010). Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

In the figure, the system-of-interest (SoI) goes into operation as a delivered product or offered service in the utilization stage while maintenance and logistics are provided (by a product sustainment system) simultaneously in the support stage. These two stages are commonly executed in parallel, and they offer opportunities to observe any need for changes in the properties of the product or service or how it is operated and supported. Making changes iterates the life cycle and results in new or improved products or features.

The delivered product and its enabling systems collectively form a wider system-of-interest (WSOI). The project design enabling system is an enterprise based system asset that establishes the strategy and means of organizing the

projects to be executed along the life cycle. In many larger organizations, this type of enabling system is institutionalized and can be based upon recommendations of the Project Management Institute (PMI).

Product systems should be viewed as enabling service systems. That is, once deployed, a product system provides a service that contributes to an enterprise's operations. To the acquirer, the SoI provides operational services to users. This is true at several levels:

- Hardware and software products are used to enable the provisioning of service systems,
- Enterprises incorporate products as system assets and use them to enable operations, and
- Provided products are integrated into the system of systems.

Product Architecture, Modeling, and Analysis

IEEE standard 1471-2000 defines architecture as "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" (IEEE 2000). Similarly, ISO/IEC 42010-2011 defines architecture as "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" (ISO/IEC 2011).

A product's purpose (stakeholder's need) is realized by a product system (the SoI). Because product systems are composed of different entities (components, assemblies, subsystems, information, facilities, processes, organizations, people, etc.) that together produce the results unachievable by any of the entities alone, architecting the product is based on a whole systems approach. To architect with a whole systems approach means to define, document, design, develop, manufacture, distribute, maintain, improve, and to certify proper implementation of the product's objectives in terms of the functional (the "what"), the behavioral (the use, or intended operations), the logical (interaction and relationships between entities) and the physical constructs (Wasson 2006; Maier 2009; Blanchard and Fabrycky 2011).

The system architect starts at the highest level of abstraction, concentrating on needs, functions, systems characteristics and constraints (concerns) before identifying components, assemblies, or subsystems. This is the systems view, and it is used to represent the stakeholder's market service description or the concept of operations (understanding of the opportunity/problem space).

Next to be documented, as needs become better understood, are architectural descriptions at different levels of abstraction, representing various stakeholders interests. These are the architecture models. They define the possible solution spaces for the product purpose in the form of detailed system, operational, behavioral, and physical requirements of the product system.

Different modeling techniques are then used to analyze different types of requirements. For operational scenarios and different modes of operation, there are hierarchical decomposition and allocation, architectural block diagrams (ABD), functional block diagrams (FBD), functional flow block diagrams (FFBD), and use case diagrams. For interactions and relationships among hardware and/or software components there are sequence diagrams, activity diagrams, state diagrams, and data flow diagrams. See (Maier 2009) Chapter 8 for an introduction to models and modeling.

Analysis of the solution space makes it possible to produce detailed technical specs, engineering drawings, blueprints, software architectures, information flows, and so on, that describe the entities in the product system. An entity's requirements bound its attributes and characteristics, levels of performance, operational capabilities, and design constraints. During design and integration, entity characteristics can be traced back to requirements (requirements traceability being a key aspect of SE). Verification and validation plans created during the requirements phase are the basis of testing certification that the product does what it was intended to do.

Overall, what occurs is the transformation of a set of requirements into products and processes that satisfy the stakeholder's need. The architecture is represented by a set of models that communicate an integrated view of the

product's intent and purpose, and the interactions and interfaces required among all the different participating entities. The product's purpose is articulated in terms of business objectives (market, cost, functionality, performance, and time to deliver). The set of models includes sufficient variety to convey information appropriately to the stakeholders, designers/developers, specialty engineering, operations, manufacturers, management, and marketing and sales personnel.

Different architecture frameworks have been developed to guide product teams in defining the product architecture for commercial and for public enterprises. In general, an architecture framework describes a "view," meaning a "collection of models that represent the whole system from the perspective of a set of related stakeholder concerns." Prime examples of architecture frameworks are the Zachman framework (Zachman 1992), The Open Group Architecture Framework (TOGAF) (TOGAF 2011), the Enhanced-Telecom Operations Map (e-TOM), just to mention a few in the commercial sector. In the case of public enterprises a few architecture frameworks include the Department of Defense Architecture Framework (DODAF 2.0) (DoD 2010), the Federal Enterprise Architecture Framework (FEAF) (FEA 2001), the British Ministry of Defense Architecture Framework (MODAF) (MOD 2004), etc.

Differences between acquired products and offered products play an important role in defining product system requirements. Acquired products are life cycle-managed directly by the acquirer; for instance, acquired defense systems are defined, developed, tested, owned, operated, maintained and upgraded by the defense agency. See the article Product Systems Engineering Key Aspects within this KA.

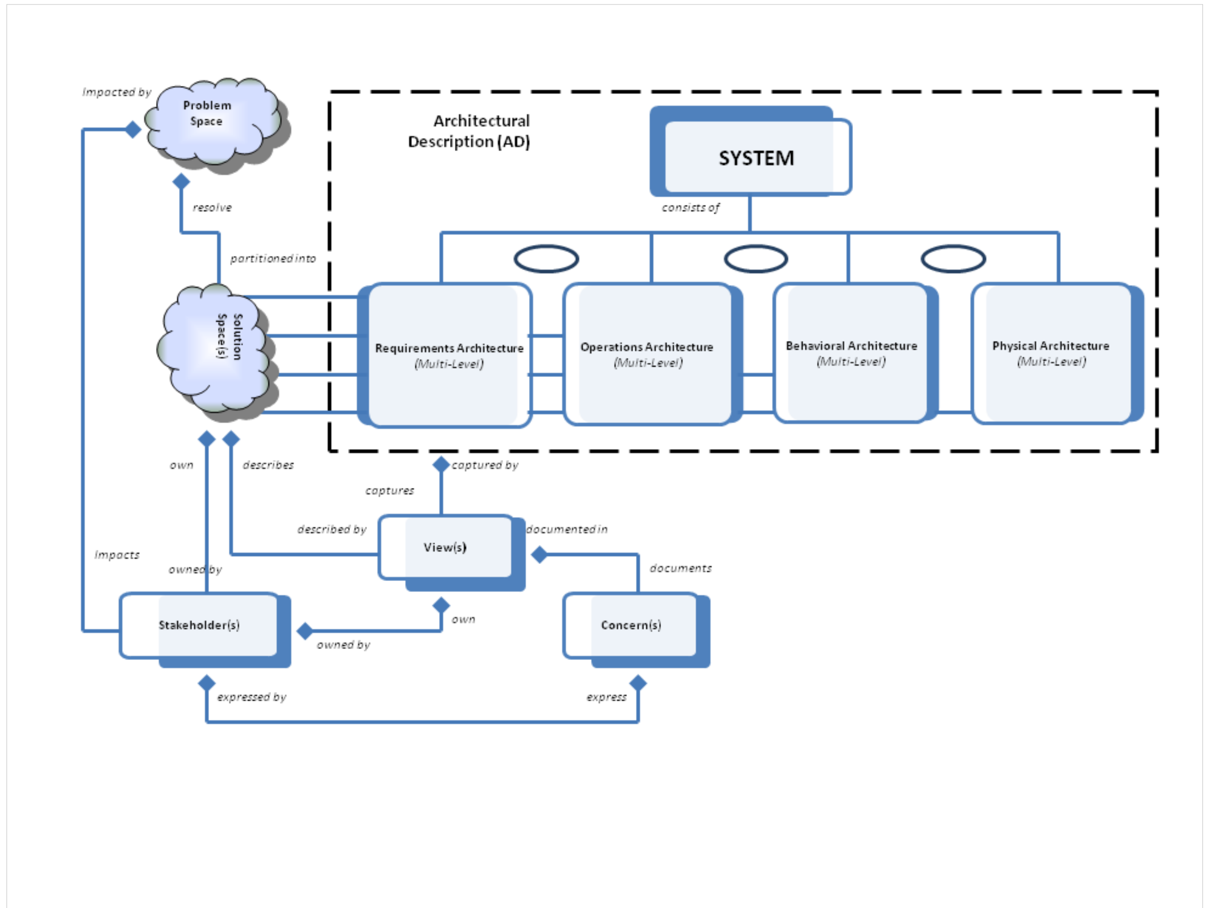


Figure 2. System Architectural Description Elements (Adapted from Wasson 2006). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Specialty Engineering Integration

The INCOSE *SE Handbook* defines specialty engineering as:

"Analysis of specific features of a system that requires special skills to identify requirements and assess their impact on the system life cycle."

Areas of expertise that fall under this umbrella definition include logistics support, electromagnetic compatibility analysis, environmental impact, human factors, safety and health analysis, and training. The unique characteristics, requirements, and design challenges of a system-of-interest all help determine the areas of specialty that apply.

A number of specialty engineering areas are typically important to systems engineers working on the development, deployment, and sustainment of product systems. For example, logistics support is essential for fielded product systems that require maintenance and repair. The delivery of services, materials, parts, and software necessary for supporting the system must all be considered very early in the development activity. These factors should usually be considered before the system requirements and concept definition are complete. To integrate these specialty disciplines sufficiently early on, the systems engineer needs to know what specialties relate to the system under development, how they relate to the systems engineering process, and how to integrate them into the life cycle process.

For product systems with significant hardware content and that operate in challenging environments, the following specialty engineering areas must usually be considered:

- manufacturability,
- reliability and maintainability,
- certification (essential where human safety is an issue),
- logistics support,
- electromagnetic compatibility (if they radiate),
- environmental impact,
- human factors,
- safety and health, and
- training.

The relationship of these specialty areas to the systems engineering process must be understood and considered. The key aspects of the relationship are:

- when the specialty needs to be considered,
- what essential data or information it provides,
- the consequences of not including the specialty in the systems engineering process, and
- how the systems engineers should interact with the specialty engineers.

Grady (2006) provides an overview, with references, for many of the specialty engineering disciplines, including reliability engineering; parts, materials, and process engineering (PMP); maintainability engineering, availability, producibility engineering, design to cost/life cycle cost (DTC/LCC), human factors engineering, corrosion prevention and control (CPC), system safety engineering, electromagnetic compatibility (EMC) engineering, system security engineering, mass properties engineering, and environmental impact engineering.

Eisner (2008) lists specialty engineering as one of the "thirty elements" of systems engineering. "Specialty engineering refers to a set of engineering topics that have to be explored on some, but not all, systems engineering efforts. In other words, some systems involve these special disciplines and some do not. Examples of specialty engineering areas include electromagnetic compatibility and interference, safety, physical security, computer security, communications security, demand forecasting, object-oriented design, and value engineering." Some of what we consider specialty engineering in the present article, Eisner includes among his "thirty elements" of systems engineering, but not as part of the specialty engineering element.

There is no standard list of specialty engineering disciplines. What is considered specialty engineering varies according to the community to which the systems engineering belongs, and sometimes to the preferences of the customer.

References

Works Cited

ANSI/IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY, USA: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

Eisner, H. 2008. "Chapter 7. Essentials of Project and Systems Engineering Management," in *The Thirty Elements of Systems Engineering*, 3rd ed. New York, NY, USA: John Wiley & Sons.

Grady, J. 2006. "Chapter 3.7. System Requirements Analysis," in *Specialty Engineering Requirements Analysis*. New York, NY, USA: Academic Press.

Grady, J. 2006. *System Requirements Analysis*. New York, NY, USA: Academic Press.

Grady, J. 2010. *Systems Synthesis - Product and Process Design*. Boca Raton, FL, USA: CRC Press.

Hybertson, D. 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boston, MA, USA: Auerbach Publications.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications.

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting*, 3rd ed. Boca Raton, FL, USA: CRC Press.

MOD. 2004. *Ministry of Defence Architecture Framework (MODAF)*, version 2. London, UK: UK Ministry of Defence.

The Open Group. 2011. *TOGAF*, version 9.1. Hogeweg, The Netherlands: Van Haren Publishing. Accessed August 29, 2012. Available at: https:/ / www2. opengroup. org/ ogsys/ jsp/ publications/ PublicationDetails. jsp?catalogno=g116.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

Zachman, J.A. 1992. "Extending and Formalizing the Framework for Information Systems Architecture." *IBM Systems Journal*. 31 (3): 590-616.

Primary References

Eisner, H. 2008. "Chapter 7. Essentials of Project and Systems Engineering Management," in *The Thirty Elements of Systems Engineering*, 3rd ed. New York, NY, USA: John Wiley & Sons.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

Additional References

ANSI/IEEE. 2000. *Recommended practice for architectural description for software-intensive systems.* New York, NY, USA: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

Grady, J. 2006. "Chapter 3.7. System Requirements Analysis," in *Specialty Engineering Requirements Analysis*. New York, NY, USA: Academic Press.

Grady, J. 2006. *System Requirements Analysis*. New York, NY: Academic Press.

Grady, J. 2010. *Systems Synthesis- Product and Process Design*. Boca Raton, FL, USA: CRC Press.

Hybertson, D. 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boston, MA, USA: Auerbach Publications.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC. 2008. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2008.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting*, 3rd ed. Boca Raton, FL, USA: CRC Press.

Zachman, J. 2008. "John Zachman's Concise Definition of The Zachman Framework™." Zachman International Enterprise Architecture. Accessed August 29, 2012. Available at: http:/ / www. zachman. com/ about-the-zachman-framework.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Business Activities Related to Product Systems Engineering

This topic discusses the interfaces between product systems engineering and other 'back office' and management activities that take place in an enterprise.

Marketing, Product Life Cycle Management, & Quality Management

Product systems engineering (PSE) includes critical and robust interfaces with related business activities, such as marketing, product life cycle management (PLM), and quality. Traditionally, PLM has been a critical stage in the integrated product development process (IPDP) and continues to be an important tool for life cycle management after product deployment. PLM provides an important component of the PSE end-to-end view. The other component is the “breadth” component that captures everything relevant to the system at each life cycle stage. Recently, the focus has started to shift from the idea of managing just the life of the product, to an expanded view that includes the management of product-lines (families) or product platforms themselves. This provides an increase in sustainability, flexibility, reduced development times, and important reductions in costs as new or enhanced products are not launched from scratch every time.

PSE also includes interfaces with the marketing function; in particular, PSE works closely with the business and market development organizations to elicit product needs and intended operations in target markets to define product roll-out and possible phases of product introduction. Analysis of the market is critical during the entire product life cycle from conception through retirement with the understanding that each life cycle phase requires very different marketing approaches. During concept development, marketing has to help determine the potential market, the addressable market segments, define products, and product/innovations requirements for those markets. During the product introduction stage, marketing has to create demand and prompt early customers to try the product. During the growth and maturity phases, marketing has to drive public awareness, develop the brand, and differentiate the product and its features and feature releases to compete with new market entrants. During saturation, marketing must help manage diminishing volumes and revenues as focus shifts from top line (increased market share) to bottom line (increased production and distribution efficiencies) considerations. See the article on Procurement and Acquisition.

The links between PSE and quality are just as critical. The relationships between PSE and quality also reflect the broad view which includes the product and opportunity, but also the company’s internal goals, processes, and capabilities. Quality schemes which focus on a tangible product have been extensively used historically. More recent approaches that acknowledge and match PSE's holistic view have come into use. Issued during 1988, ISO 9000 is a family of standards which focuses on processes and the organization instead of the product itself. In addition, it calls out specific requirements for the design of products and services. ISO 9001 has served as a “platform” for many other schemes which are tailored to specific domains. A collaborative effort of the International Aerospace Quality Group, AS9100 contains all of the base requirements of ISO 9100 and expands further requirements which are critical and specific to the aviation, space, and defense industries. Similarly QS-16949 is a technical standard based on ISO 9001 but expanded to meet specific requirements in the worldwide automotive industry. PSE should play an important role in the design and implementation of any quality management system. See the article on Quality Management.

Capability Maturity Model Integrated (CMMI) for Development is a process improvement approach whose goal is to help organizations improve their performance. CMMI can be used to guide process improvement across a project, a division, or an entire organization. Although initially used in software engineering and organizational development, CMMI use is spreading to other domains since it provides organizations with the essential elements for effective process improvement. According to the Carnegie Mellon Software Engineering Institute, CMMI describe "an evolutionary improvement path from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness." (SEI 2010).

Project Management & Business Development

The end-to-end view mandated by PSE requires strong relationships with project management and business development activities. The ‘concurrent’ thinking encouraged by PSE necessarily requires multiple projects to move forward in parallel, but with a high level of coordination. In this sense, PSE and project management (see Systems Engineering and Project Management) are two heavily intertwined disciplines which have been shown to generate synergy and added value for the stakeholders.

The systems engineering management plan (SEMP) is the key document covering the activities, milestones, organization, and resource requirements necessary to ensure the system-of-interest accomplishes its intended goals and mission. A key objective of the SEMP, which is usually developed during the conceptual design phase, is to provide the structure, policies, and procedures to foster the optimum management of projects and activities needed for system development and implementation (Blanchard and Fabrycky 2011).

An effective and agile PSE function can make important contributions to business development for an enterprise or company. The primary goal of business development activities is to identify new types of business/product/services which are believed to address existing or potential needs and gaps (new markets), to attract new customers to existing offerings, and to break into existing markets. PSE’s end-to-end view of the life cycle can support market development by intelligence gathering, feedback on market acceptance or rejection, strategic analysis, and proposition development and campaign development. Finally, PSE should encourage the consideration of several factors within the new product development which may enhance market development. For example, in well-established companies, business development can often include setting up strategic alliances with other, third-party companies. In these instances, the companies may leverage each others expertise and/or intellectual property to improve the probability for identifying, researching, and bringing to market new businesses and new products. See (Sørensen 2012).

Supply Chain Management & Distribution Channel Management

PSE provides the following information to the supply chain management function in an enterprise:

- product specifications (including intended uses of the product),
- product acceptance criteria (for accepting delivery of the product from the supplier),
- product testing and qualification plans and procedures, including which ones are responsibility of the supplier and which ones are responsibility of the acquirer,
- interface specifications associated with each product,
- supplier certification criteria (including a list of pre-certified suppliers), and
- feedback on quality of products delivered by suppliers.

Supply chain management will, as necessary, manage the identification and certification of qualified suppliers with the concurrence of, and coordination with, systems engineering and product engineers.

PSE provides the following information to the distribution channel management function in the enterprise:

- product specifications (including intended uses of the product),
- product user manuals (including installation and maintenance documentation),
- product packaging (for safe delivery of product and for display in retail channels),
- product qualification data (to prove that product meets its design requirements),
- product certification data (to prove product is certified for safe and secure operation),
- user support instructions, and
- operator certification criteria.

Distribution channel management will, as necessary, manage the identification and certification of qualified distributors with the concurrence of, and coordination with, systems engineering and product engineers.

Capability Management & Operations Management

Capability is defined in various ways, but each definition is consistent with the notion of "the ability to do something useful." Products and services are acquired by end users to enable and improve their operational capability to let them do something useful, whether in a military context (e.g., weapon systems improve the capability to conduct effective military operations), or a social context (e.g., a car may improve the ability to satisfy the transport needs of a family). Users acquire products (e.g., military equipment, cars, “productized” service offerings from airlines and taxi companies, etc.) to contribute to satisfying their capability needs.

Capability management involves identifying and quantifying capabilities (existing, new, or modified) that will be needed to meet enterprise goals, as well as selecting a coherent set of product and services across all components of Capability (glossary) that will be integrated to provide the needed capabilities. So normally, requirements for "product systems" are derived from capability management. Capability management is likely to include trade-off processes to make the best use of existing products or low-risk evolutions of them, and conversely identifying when a capability need can only be satisfactorily met by a new-generation product, or even a new type of product altogether. In some cases, new offered products or disruptive technologies (e.g., jet engine, nuclear weapons, and internet) create opportunities for new or improved capabilities, in which case capability management focuses on ensuring that all needed components of capability are put in place to exploit the opportunity provided by the new product or technology. See Capability Engineering.

Operations management uses an integrated set of product systems to deliver value to the enterprise and its stakeholders. Operations management involves bringing new product systems into operation, normally while maintaining business continuity, so transition plans and relevant metrics are critical; next, operations management addresses some of the following: working up to full operational efficiency across all components of capability, coping with incidents, contingency plans to deal with major disruptions, adjusting the system to cope with new ways of working and to deliver new services to meet new enterprise requirements and accommodate new product systems entering service, and eventually planning transitions out of service or major in-service upgrades. PSE supports operations management by defining all dependencies for successful operation on other systems and services, and by providing ongoing engineering support for spares and repairs, obsolescence management, and system upgrades. Systems engineering in the in-service phase has been analyzed (Elliott et al. 2008) and is best viewed as the same basic systems engineering process conducted at a much higher tempo (Kemp and Linton, 2008) and requiring detailed understanding of constraints imposed by the current environment and usage. Configuration management and configuration status accounting during operation is very important for high value and high integrity systems to ensure that any changes are designed to fit the "as-is" system, which may be significantly different from the "as-originally intended" specification and design.

Product Engineering, Assembly, Integration, & Test

Product engineering typically results in an engineering model that is used as the “blueprint” for assembling, integrating, and testing (AIT) a product system. These AIT activities may be performed on prototype versions, as well as final production versions to be delivered to end users. There is significant experience in domain specific industries in performing AIT for complex products. Unfortunately, very little is written in the general literature. Wasson (2006) and de Jong (2008) cover some of these aspects. See also System Integration and System Verification.

For software products, the collection of code modules are integrated via some form of integration program (typically called “make”). The integrated modules are then subjected to tests to exercise the various potential paths through the software. Since software can be easily changed, it is common to use some form of regression testing based upon test suites in order to verify software correctness. Another common means of testing is by fault injection as described by Voas and McGraw (1998).

Manufacturing, Test, & Certification

Systems engineers usually work with manufacturing indirectly through the electrical and mechanical design teams. There are times in the development cycle when a direct interface and working relationship between systems engineering and manufacturing is appropriate and can improve the probability of program and system success. Early in the program the system concept must be examined to determine if it is manufacturable. The requirements and the concept design should be reviewed with the manufacturing engineers to obtain an assessment of the risks associated with the production of the system. If substantial risks are identified, then actions that improve the manufacturing capabilities of the organization, modify the design, and perhaps change the requirements may be needed to reduce the identified risks to acceptable levels. Manufacturing prototypes or proof of manufacture (POM) units may be necessary to reduce the risk and to demonstrate readiness to proceed with the design and the system development. Similarly, the systems engineers must establish that the system will be testable early in the product development phase. The requirements should be mapped to verification methods of inspection, analysis, demonstration, and test before they are released to the design team. All requirements mapped to test must be examined to determine the test methods and the risk associated with accomplishing the necessary tests as part of the product qualification, acceptance, and release process. Where risks are identified, the systems engineers must work with the test engineers to develop the necessary test capabilities.

Product Delivery & Product Support

Most products live much longer in the usage phase than in the development phase. The costs associated with product support are usually greater than the cost of developing the product. These two facts make it very important for the product systems engineer to consider the product delivery and support as part of the earliest activities during development. The design of the product dictates the maintenance and support that will be required. The systems requirements are the first means of influencing the design to achieve the desired product support. If maintenance, reliability, and support requirements have not been defined by the customer, then the systems engineer must define these to achieve the support methods and costs that the customer, users, and the organization responsible for support will find financially acceptable.

References

Works Cited

de Jong, I. 2008. *Integration and Test Strategies for Complex Manufacturing Machines: Integration and Testing Combined in a Single Planning and Optimization Framework*. Saarbrücken, Germany: Verlag.

Elliott, B. et al. INCOSE UK Chapter Working Group on Applying Systems Engineering to In-Service Systems, Final Report. Somerset, UK: INCOSE UK Chapter Working Group. 2008. Accessed November 11, 2014 at INCOSE UK http://www.incoseonline.org.uk/Documents/Groups/InServiceSystems/is_tr_001_final_report_final_1_0.pdf.

Kemp, D., and R. Linton. 2008. "Service Engineering." Proceedings of the 18th Annual International Symposium of the International Council on Systems Engineering, June 15-19, 2008, Utrecht, The Netherlands.

CMMI Product Team. *CMMI for Development Version 1.3* (CMU/SEI-2010-TR-033). 2010. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. Accessed on 10 Nov 2014 at Software Engineering Institute Library http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661

Sørensen, H.E. 2012. *Business Development: a market-oriented perspective*. Hoboken, NJ, USA: John Wiley & Sons.

Voas, J.M., and G. McGraw. 1998. *Software Fault Injection*. Hoboken, NJ, USA: John Wiley & Sons.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

Primary References

de Jong, I. 2008. *Integration and Test Strategies for Complex Manufacturing Machines: Integration and Testing Combined in a Single Planning and Optimization Framework*. Saarbrücken, Germany: Verlag.

Voas, J.M., and G. McGraw. 1998. *Software Fault Injection*. Hoboken, NJ, USA: John Wiley & Sons.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

Additional References

Phillips, F.Y. 2001. *Market-Oriented Technology Management: Innovating for Profit in Entrepreneurial Times*, 1st ed. New York, NY, USA: Springer.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Product Systems Engineering Key Aspects

Acquired Products versus Offered Products

The emphasis for traditional systems engineering (TSE) is in the provisioning of products and related services that meet stakeholder needs and requirements. For acquired products, an acquirer specifies the needs and requirements, selects a supplier for development and provisioning, and then receives the needed products and services. The acquirer, after acceptance, usually owns, operates, and maintains the product and the support systems supplied by the developer. Offered products are provided by suppliers based on opportunities to develop and offer products and services to potential users of the product based on business objectives usually measured in terms of value addition to the stakeholder.

In the provisioning of product systems and related services, the enterprise owning and provisioning the product and services typically makes agreements with other suppliers to also provide elements, methods, and tools that are used during their entire life cycle. The supplying enterprises, in turn, may make further agreements with suppliers in regards to building a supply chain. The complexities of dealing with supply chains must be accounted for with respect to cost, risk, and schedule and thus can have an impact upon product or service maturity. (See articles under the Systems Engineering Organizational Strategy knowledge area (KA) in Part 5.)

Acquired Products

Specific needs for a product or service typically result in some form of an agreement between the acquirer and a supplier as specified in the agreement processes of ISO/IEC 15288 (2015). The acquirer specifies the need and requirements for the properties of the expected product or service and may or may not place specific requirements upon how the supplier plans to organize their life cycle treatment of the product or system.

The degree of formality involved with the agreement varies and is strongly influenced by whether the customer is a government entity or a commercial entity. Government contracts usually incorporate strict specifications and other unique requirements that are rarely found in commercial contracts. Government acquisition agents often specify design characteristics in addition to functional and performance specifications. Design specifications place constraints on product systems engineering (PSE) by explicitly defining the details of a product's physical characteristics. The government acquirer may also specify how the product is to be designed and developed or how it is to be produced. Government specifications tend to be longer, more detailed, and more complex than functional specifications and much longer than specifications used in a commercial environment.

When contracting with the government or similar enterprises, the PSE must identify disagreements related to the meaning of a particular provision in a contract, and work with contracts to get a written resolution of all ambiguities and issues in the specifications. Failure to do this can lead to legal disputes and government claims of product substitution which can prevent acceptance of the product system and result in financial penalties.

Developing product systems for government customers requires PSE to do a thorough review and perform internal coordination within the enterprise to prevent it from submitting proposals that are non-compliant because the requirements are not fully understood.

Offered Products

Given an opportunity or perceived opportunity, an enterprise may decide to develop and offer products or services to a broader potential marketplace. The properties of the product or service are often determined through surveying and/or forecasting the potential market penetration. The supplier determines the structure and operation of an appropriate life cycle model for achieving the desired results (Pugh 1990).

Supply Chains and Distribution Channels

The supply of products and services to the owner of a product or service that is acquired or offered at various points during the life cycle is vital to success. It is this wider system-of-interest (WSOI) that is the outsourcing holism that must be treated properly in order to provide successful products or services. A portrayal of supply chain structure is provided in Figure 1 below.

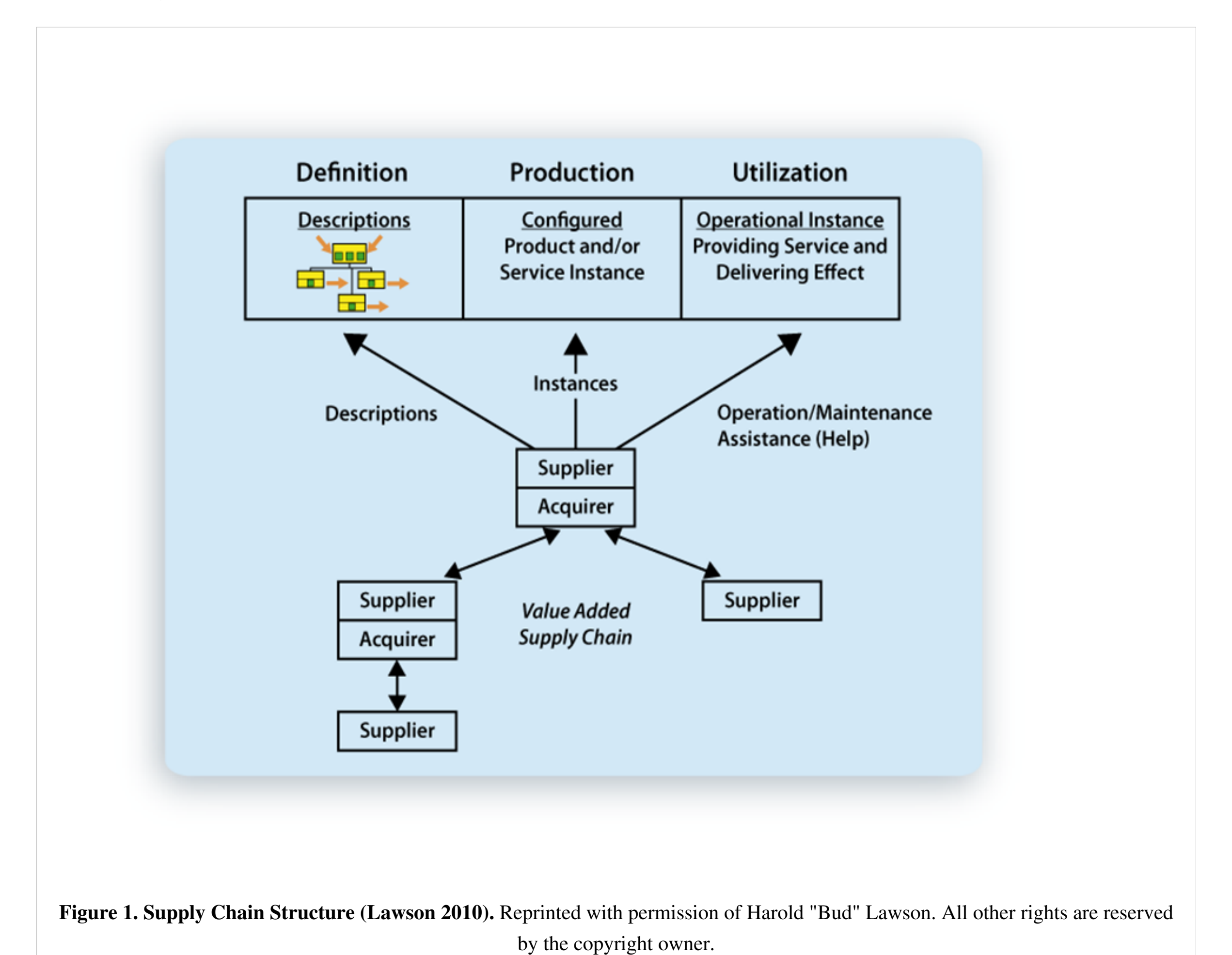


Figure 1. Supply Chain Structure (Lawson 2010). Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

In Figure 1, it is important to note that in an agreement with a supplier, the outsourcing can involve delivering complete system description solutions or portions thereof. For example, a supplier could, given a set of stakeholder

requirements developed by the acquirer, develop and supply a system that conforms to the architectural solution. The supplier in turn can be an acquirer of portions of their delivered results by outsourcing to other suppliers.

In regards to production, the outsourcing agreement with a supplier can vary from total production responsibility to merely supplying instances of system elements to be integrated by the acquirer. Once again, these suppliers can be acquirers of portions of their delivery from outsourcing to other suppliers.

In regards to utilization, for non-trivial systems, outsourcing agreements can be made with a supplier to provide for operational services, for example, operating a health care information system. Further agreements with suppliers can involve various forms of logistics aimed at sustaining a system product or service or for supplying assistance in the form of help desks. Once again, suppliers that agree to provide services related to utilization can be acquirers of the services of other suppliers.

Important to all supply chains is the concept that supplying parties contribute some form of added value to the life cycle of a system-of-interest. The proper management of a supply chain system asset is a vital part of the operations of an enterprise. In fact, the supply chain itself is an enterprise system-of-interest that is composed of acquirers and suppliers as system elements. There is definitely a structure tied together by agreement relationships. Further, the operation of the supply chain results in an emergent behavior. The supply chain system becomes a vital infrastructure asset in the system portfolios of enterprises and forms the basis for extended enterprises.

Similar to a supply chain, the distribution channels for a product system can be a complex web of relationships between the product supplier and various distributors, for example, package delivery companies, warehouses, service depots, wholesale outlets, retail sales establishments, operator training and certification organizations, and so on. The nature of the distribution channels could have a significant impact on the architecture or design of a product system.

PSE may need to include special features in the product design to accommodate for the needs of distribution channel elements, for example, heavy load tie down or lifting brackets, protective shipping packages, retail marketing displays, product brochures, installation manuals, operator certification packages, training materials, and so on. Sometimes it may be necessary to create special versions (or instances) of the product for the training of operators and users for certifying safe or secure operations, for environmental testing and qualification, for product demonstration and user testing, for patent application, for load testing and scalability demonstrations, and for interface fit checking and mass balance certification, to name some examples.

Product Lifecycle and Product Adoption Rates

The life cycle of each product follows the typical incremental development phases shown below (Wasson 2006, 59-65). A particular product to be engineered could be preceded by a previous "model" of that product as shown in the product model life cycle below, and could be superseded later by a newer model of that product. It is worth noting that there is no standard set of life cycle phases. The example below is one of many ways that the phases can be structured.

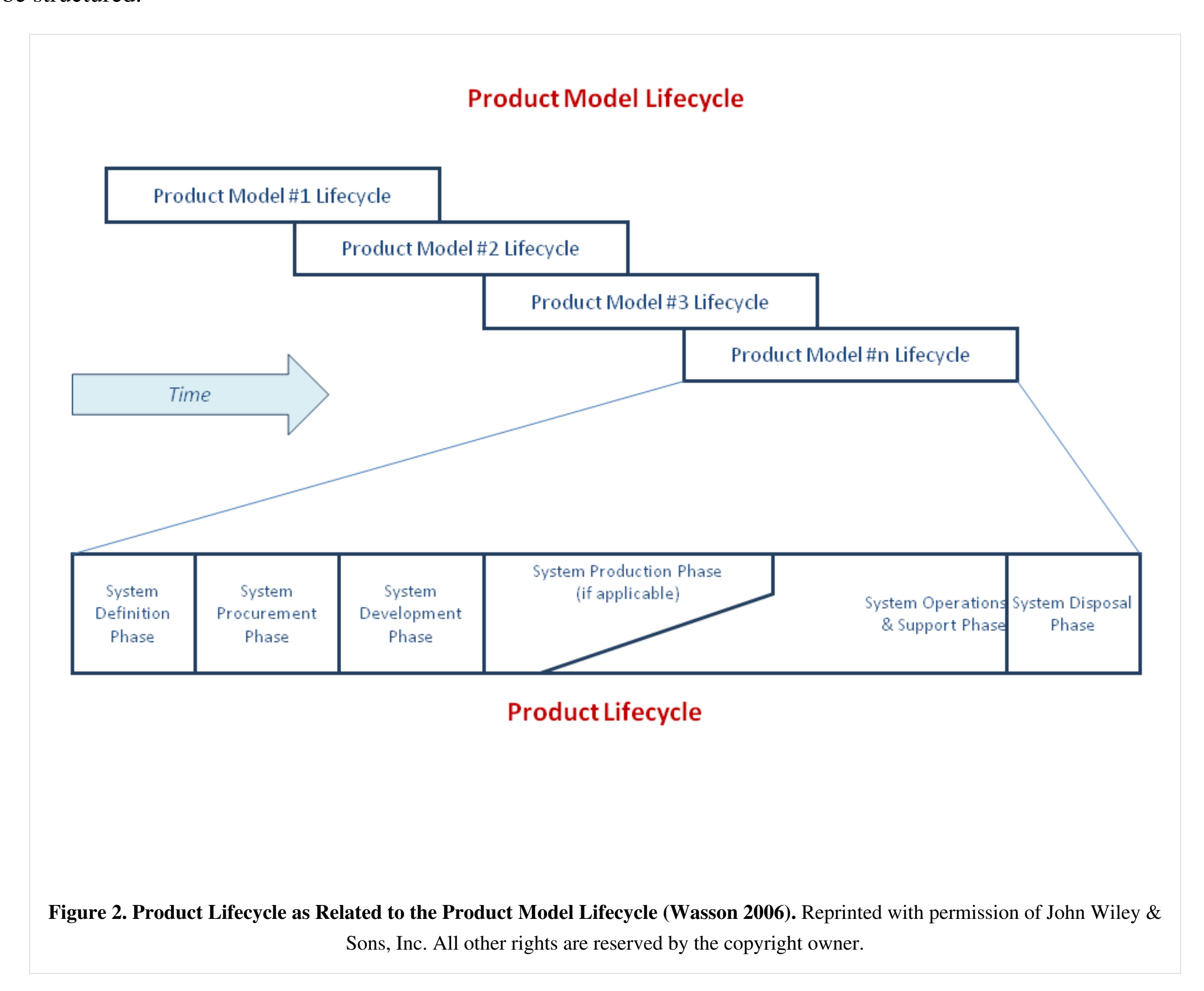


Figure 2. Product Lifecycle as Related to the Product Model Lifecycle (Wasson 2006). Reprinted with permission of John Wiley & Sons, Inc. All other rights are reserved by the copyright owner.

From an industry perspective, managing a product's life cycle involves more than just the engineering aspects:

> *Product lifecycle management (PLM) is the process of managing the entire lifecycle of a product from its conception through design and manufacture to service and disposal. PLM integrates people, data, processes and business systems, and provides a product information backbone for companies and their extended enterprise.* (CIMdata 2012)

There are many PLM tools and services available for facilitating the development and management of complicated product life cycles and especially for product line management (insert link to product line mgmt section here).

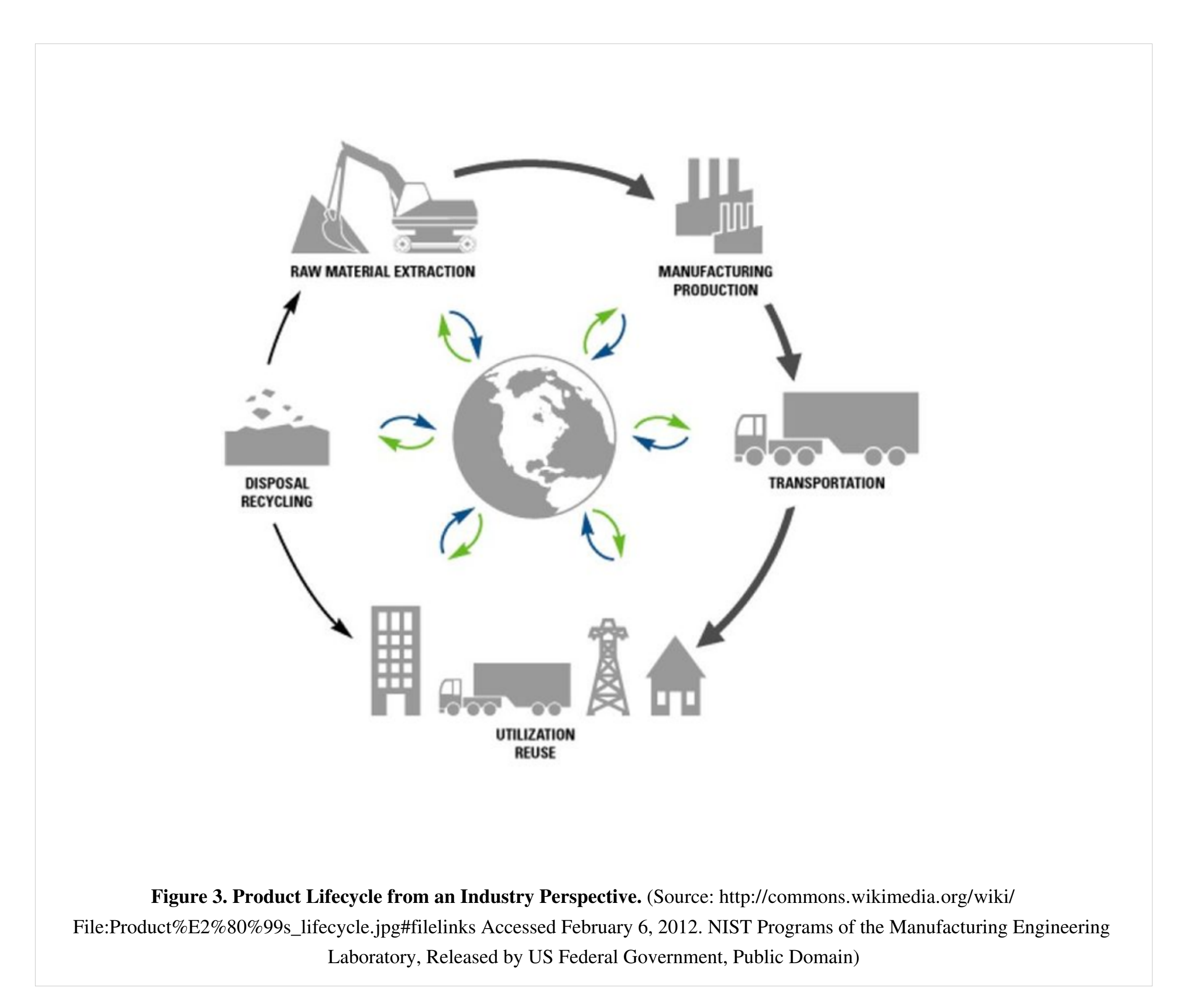


Figure 3. Product Lifecycle from an Industry Perspective. (Source: http://commons.wikimedia.org/wiki/File:Product%E2%80%99s_lifecycle.jpg#filelinks Accessed February 6, 2012. NIST Programs of the Manufacturing Engineering Laboratory, Released by US Federal Government, Public Domain)

The product and product model life cycles are driven by the product adoption rate, illustrated below, that is commonly experienced by most engineered products (Rogers 2003). As products reach market saturation (i.e., on the down slope of the curve below) then there would typically be a new, upgraded version of the product ready for delivery to the marketplace. PSE serves a critical role in determining the best timing for delivery of this new version and the set of features and functions that would be of the greatest value at that time.

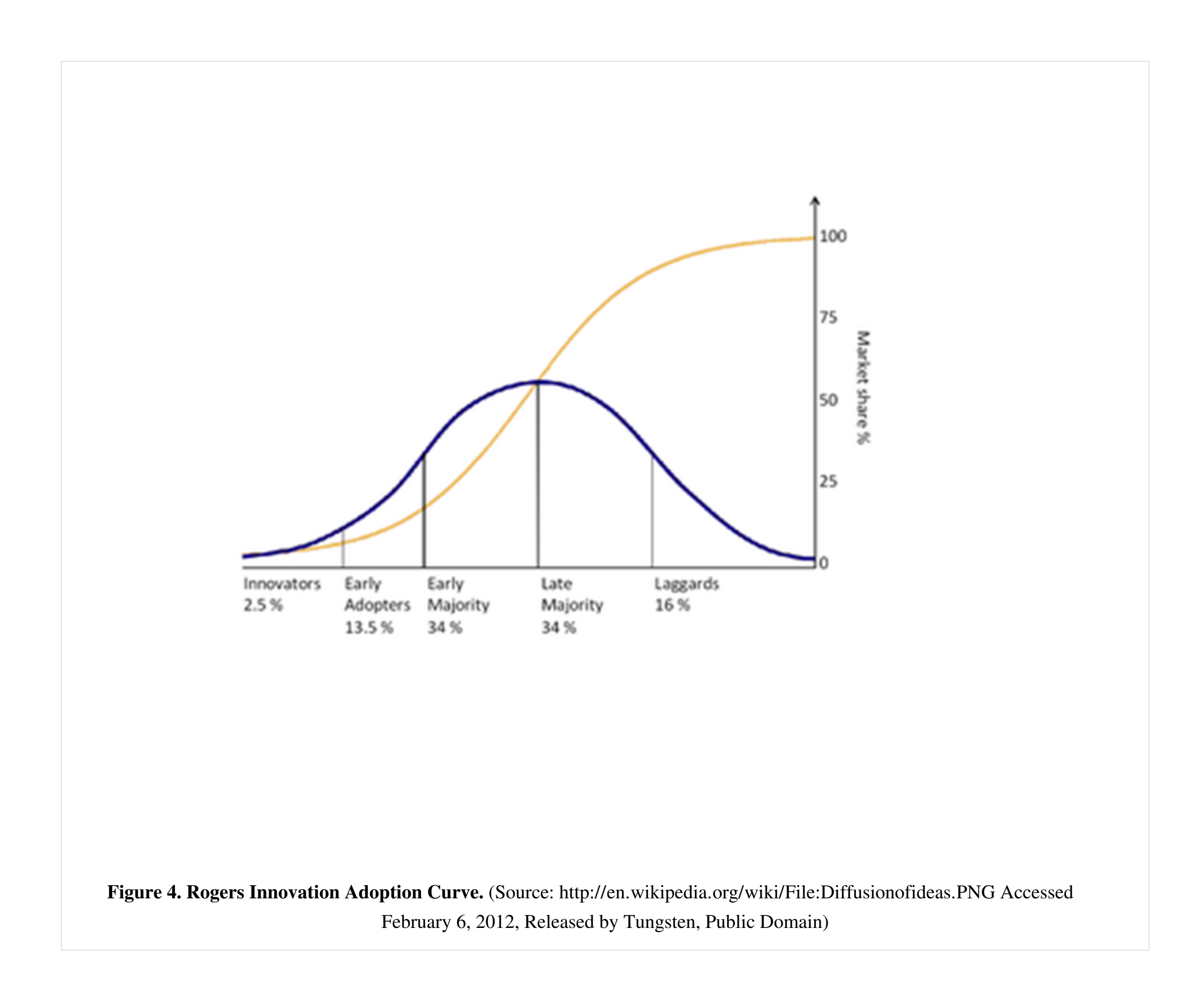


Figure 4. Rogers Innovation Adoption Curve. (Source: http://en.wikipedia.org/wiki/File:Diffusionofideas.PNG Accessed February 6, 2012, Released by Tungsten, Public Domain)

Integrated Product Teams and Integrated Product Development

Product systems as discussed throughout this KA mandate the participation of different disciplines for their success during their entire lifecycle from concept to product disposal or retirement. Rapid technology innovations and market pressures in the mid '90s demanded development process (mostly input-output serial) to shorten their development time and development cost, and to improve product quality to remain competitive. For commercial enterprises, the typical development times of 18-24 months to deploy new products into markets of the '90s have in many cases been reduced to 6-12 months and even 3-6 months for the highly competitive leading edge information technology products.

An initial response to these pressures was concurrent engineering. Concurrent engineering is “... a systematic approach to the integrated, concurrent design of products and their related processes, including manufacturing and support to cause developers, from the outset to consider all elements of the product lifecycle from conception through disposal, including quality, cost, schedule and end user requirements." This definition has evolved into the integrated product development (IPD) as more descriptive of this concurrency to describe the continuous integration of the entire product team, including engineering, manufacturing, test, and support through the life cycle. Later, as the importance of the process was recognized, the terminology was modified to integrated product and process development or IPPD (INCOSE 2012).

The INCOSE *Systems Engineering Handbook* v. 3.2.2 provides a good description of the IPT and IPDT process; the different types of IPDT; the steps in organizing and running an IPDT; good examples of IPDT, particularly for acquired systems; and a good discussion on IPDT pitfalls to avoid. (INCOSE 2012)

IPD/IPPD helps plan, capture, execute, and evaluate programs to help design, test, build, deliver, and support products that satisfy diverse stakeholder requirements. IPD/IPPD outlines the necessary infrastructure needed to deploy, maintain, evaluate and continuously improve processes and tools by aligning people (IPTs) and processes to realize product goals (customer satisfaction). The implementation of Integrated Product Development Processes (IPDP) requires an integrated approach for program planning and generally includes the following: Business Strategy, Program Management and Control, Project Planning, Product Requirements and Architecture Development, Product Design and Development, Production and Deployment, Product Verification and Validation, and Operations and Maintenance Support.

At each development stage, there is a decision gate that helps decide if the IPDP is feasible to enter the next stage of product development. IPD utilizes multi-functional IPTs to optimize the individual product and processes to meet overall cost and performance objectives. IPTs are a cross-functional group of people typically including representatives of all the relevant stakeholders in the project, who are brought together for delivering an integrated product to an external or internal customer using relevant IPDP. The main function of the IPTs is to ensure the business, technical and economical integrity and overall success of the product that is delivered to its eventual customer. IPTs carry out tailored IPDPs and follow relevant SE processes to deliver products that satisfy customer needs, overcomes external constraints, and adheres to the overall program strategy.

In the case of commercial enterprises, product development is tightly coupled with business strategies (short and long term), stakeholder value added measured in terms of return on investments (ROI), market presence/coverage, and other strategies as defined by the business objectives. Thus, product integration teams include strategic planners, business managers, financial managers, market managers, quality assurance managers, customer representatives, and end-users, as well as other disciplines required for acquired products. Phillips (2001), Annachino (2003), and Morse (2007) provide good discussions on this topic.

Role of Architectures, Requirements, and Standards

The architectural properties of a product system are influenced by the concerns of the various stakeholders as indicated in the ISO/IEC 42010 standard (ISO/IEC 2011). The stakeholders have various views that they express based on their specific perspective. These views are vital in establishing requirements and are inputs to those responsible for defining the functions, structures, and relationships needed to achieve the desired product or service.

A number of stakeholders have been identified in the discussions of product systems. It would be possible to identify a set of important stakeholders based on the life cycle thinking provided by the ISO/IEC 15288 standard (2015), for example, one such set could consist of owners, conceivers, developers, producers, users, and maintainers as discussed by Lawson (2010). As mentioned earlier, these stakeholders should cooperate at all stages of the life cycle in specifying requirements, verifying that the requirements are met, and validating that the products produced provide needed capabilities.

In addition to the two standards that have been identified, there are a variety of standards related to specialty aspects of products, such as safety and security, as well as standards that are applicable for project management and life cycle considerations, such as requirements and quality management.

Role of Modeling, Simulation, Prototyping, and Experimentation

Modeling, simulation, prototyping, and experimentation are techniques that have the purpose of improving stakeholder knowledge and shared understanding about aspects of the system to de-risk system development and operation before heavy commitment of time and funds. Examples of this are found below:

- Understanding future needs: “Warfighting experiments are the heart of the Army's warfighting requirements determination process. Progressive and iterative mixes of high fidelity constructive, virtual, and live simulations using real soldiers and units in relevant, tactically competitive scenarios provide Army leaders with future operational capability insights" (US Army 2012),
- Simulation is used to predict and optimize aspects of system performance for which there are good mathematical or logical models before committing the final physical design, and also to verify and validate the system design in scenarios where physical testing is too difficult, dangerous, or expensive, for example, checking the performance envelope of military systems in a wide range of engagement scenarios where test firing thousands of rounds to get statistically valid data is clearly unaffordable, ensuring that the safety features in a nuclear power station will operate correctly in a wide range of stressing scenarios, etc.,
- Prototyping (physical and virtual) is used in a wide variety of ways to check out aspects of system performance, usability, utility, and to validate models and simulations as part of the iterative process of converging on a final design,
- In a manufacturing context, the first units produced are often “prototypes” intended to make sure the production process is working properly before committing to high rate production, and are often not shipped to end users, but used for intensive testing to qualify the design, and
- Simulation is also used extensively for training and marketing purposes. For training, an accurate model of the human machine interface and representation of the operational context allows operators to do most of their training without putting operational hours on the real system enabling them to learn emergency procedures for combat and accident scenarios in a safe and repeatable environment; for example, airline and military pilots now train mainly on simulators. System simulators of various levels of fidelity are used to familiarize customers and end users with the potential characteristics and benefits of the system, available options and trade-offs, and integration issues early in the development and acquisition process.

All of these methods use a variety of physical and mathematical representations of the system and its environment so modeling is an enabler for simulation, prototyping, and experimentation.

Increasing Role of Software in Product Functionality

An important trend in commercial products is the increasing importance of software in an increasingly wide range of products. Everything from phones, cameras, cars, test gear, and medical equipment now has essential functionality implemented in software. Software has had an increasing role in providing the desired functionality in many products. The embedding of software in many types of products accounts for increasing portions of product functionality. In tangible products such as cars, software helps improve functionality and usability (cruise control, climate control, etc.). In intangible products such as insurance, software helps in improving operational efficiency, data accessibility, etc.

The movement toward the internet of “things” where sensing and activating functions are incorporated is now starting to permeate. The use of various software products in proving service is also described in the Service Systems Engineering article.

Recent advancements in IT and software have assisted in their increased use in PSE. Although software development is already a very complex field, the role of software in the development and functionality of products is growing larger each day.

There is a need to broaden the horizons of software engineers to think of problem solving not only in software terms, but also using the systems thinking approach. For this purpose, software engineers need to be able to think critically about the problem and also the possible solutions to the problem or opportunity and its implication for business objectives.

Product Integration and Interface Control

Integration is "the set of activities that bring together smaller units of a system into larger units" (Eisner 2008). Products may consist of several systems, subsystems, assemblies, parts, etc., which have to work together as a whole to deliver the offered product’s functionalities at specified performance levels in the intended operations environment. Product integration entails not only the working together of hardware and software components, but also the organization, processes, people, facilities, and the resources for the manufacturing, distribution, maintenance, customer support, sales channels, etc. Grady (2010) groups the above information into three fundamental integration components: functional organization, product integration, and process integration.

PSE plays an important role to ensure well defined interfaces, interactions, relationships, information exchange, and processes requirements between product components. These requirements are baseline, documented, traced, verified, and validated for the end-to-end Product integration and to maintain and ensure product offering integrity during its life cycle. The systems engineering hierarchical decomposition level allows requirement definition and allocations at different levels of abstraction to define the building blocks of the product architecture; these building blocks are assigned to integrated product development teams (IPDTs) for detailed design and development. The IPDTs or the systems engineering integration team (SEIT) must interact with all involved players to generate appropriate architectural block specifications at the lower tier of development for a product’s architectural configuration and configuration tracking. As the building blocks are put together, interface requirements, information exchange, and interaction and relationships among entities are verified against the baseline. Once a configuration item has been built and tested against the baseline, test and verification at higher levels are conducted to obtain the final product configuration; the final product configuration can only be changed by a formal approval from a configuration control board (CCB). Note: the acronym CCB is often used to mean the change control board that, in addition to configuration control, makes decisions of any aspect of a project or an enterprise.

Interface agreements, specifications, and interface designs are usually documented through the interface control documents (ICD) and the interface design descriptions (IDD); in some instances, depending on the complexity of the product and the type of internal and/or external interfaces, an interface control working group (ICWG) is created to analyze and baseline changes to an interface for further recommendation to the CCB.

A configuration item (CI) may be hardware (HWCI), software (SWCI), firmware, subsystems, assemblies, non-development items, commercial off-the-shelf (COTS) items, acquirer furnished equipment, and/or processes. Please see Wasson (2006), Grady (2006), and INCOSE *SE Handbook* v. 3.2.2 for a more detailed description of configuration and interface control.

A product may experience hundreds of changes during its life cycle due to new product releases/enhancements, repair/replacement of parts, upgrades/updates in operating systems, computer infrastructure, software modules, organizational changes, changes in processes and/or methods and procedures, etc. Thus, strong mechanisms for bookkeeping and activity control need to be in place to identify, control, audit, account and trace interfaces, interactions, and relationships between entities that are required to maintain product configuration status (Eisner 2008). The product configuration and CI’s are then controlled through the configuration management process.

Configuration Management and Risk Management

Configuration management (CM) deals with the identification, control, auditing, status accounting, and traceability aspects of the product, and broadly covers the book-keeping and control activities of the systems engineering process (Eisner 2001). Any product configuration changes to the baseline (configuration item, operational baseline, functional baseline, behavior baseline) or product baseline are submitted to a configuration control board (CCB) through an engineering change request (ECR) and/or a configuration change request (CCR). The CCB then analyzes the request to understand CI impacts and the feasibility (time and cost) of authorization or rejection of change request(s). The lack of proper control and tracking of CI and product baselines may result in a loss of features, functionality, data, interfaces, etc., leading to backtracking and CI version losses which may affect the offered product. All approved changes will have to be baselined, documented, and tested for backward compatibility and to ensure compliance with the integrated product functionality. Thus, successful implementation and life cycle management of the product mandates a highly disciplined CM process that maintains proper control over the product and its components. Please see the INCOSE *Systems Engineering Handbook* v. 3.2.2 (2012) for a detailed description of the CM Process.

Risk management deals with the identification, assessment, and prioritization of technical, cost, schedule, and programmatic risks in any system. Almost all engineered systems are designed, constructed, and operated under some level of risks and uncertainty while achieving multiple, and often conflicting, objectives. As greater complexities and new technologies are introduced in modern systems, the potential of risks have significantly increased. Thus, the overall managerial decision-making process should involve an extensive cost-benefit analysis of all identified, qualified, and evaluated risks (Haimes 2008). Risk management involves the coordinated and most cost-effective application of resources to minimize, monitor, and control the probability and/or impact of all identified risks within the systems engineering process. The risk management process requires the involvement of several disciplines and encompasses empirical, quantitative, and normative judgmental aspects of decision-making. Furthermore, risk assessment and management should be integrated and incorporated within the broader holistic approach so technology management can help align the risk management requirements to the overall systems engineering requirements. Thus, the inclusion of a well defined risk management plan that deals with the analysis of risks, within the systems engineering master plan is vital for the long term and sustained success of any system (Blanchard and Fabrycky 2011).

References

Works Cited

Annachino, M. 2003. *New Product Development: From Initial Idea to Product Management*. Amsterdam, Netherlands: Elsevier.

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*, 3rd ed. New York, NY, USA: John Wiley & Sons.

Grady, J. 2006. *System Requirements Analysis*. New York, NY, USA: Academic Press.

Haimes, Y. 2008. *Risk Modeling, Assessment, and Management*, 3rd ed. New York, NY, USA: John Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of

Electrical and Electronics Engineers.ISO/IEC 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Kass, R. 2006. "The logic of warfighting experiments." DOD Command and Control Research Program (CCRP). August 2006. Accessed 23 April 2013 at http://www.dodccrp.org/files/Kass_Logic.pdf.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications.

Morse, L., and D. Babcock. 2007. *Managing Engineering and Technology*. International Series in Industrial and Systems Engineering. Upper Saddle River, NJ, USA: Prentice Hall.

Phillips, F. 2001. *Market Oriented Technology Management: Innovating for Profit in Entrepreneurial Times*. New York, NY, USA: Springer.

Pugh, S. 1990. *Total Design: Integrated Methods for Successful Product Engineering*. Englewood Cliffs, NJ, USA: Prentice Hall.

Reinertsen, D. 1997. *Managing the Design Factory: A Product Developers Tool Kit*. New York, NY, USA: Simon & Schuster Ltd.

Rogers, E.M. 2003. *Diffusion of Innovations*, 5th ed. New York, NY, USA: Free Press.

Smith, P., and D. Reinertsen. 1997. *Developing products in half the time – new rules, new tools*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

US Army. 2012. "Chapter 2, section A.4" in *Army Science and Technology Master Plan*. Accessed January 12, 2012. Available at: http://www.fas.org/man/dod-101/army/docs/astmp/c2/P2A4.htm.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

Primary References

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice Hall.

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*, 3rd ed. New York, NY, USA: John Wiley & Sons.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

Additional References

Annachino, M. 2003. *New Product Development: From Initial Idea to Product Management*. Amsterdam, Netherlands: Elsevier.

Haimes, Y. 2008. *Risk Modeling, Assessment, and Management*, 3rd ed. New York, NY, USA: John Wiley & Sons.

Kass, R. 2006. "The logic of warfighting experiments." DOD Command and Control Research Program (CCRP). August 2006. Accessed 23 April 2013 at http://www.dodccrp.org/files/Kass_Logic.pdf.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications.

Morse, L., and D. Babcock. 2007. *Managing Engineering and Technology*. International Series in Industrial and Systems Engineering. Upper Saddle River, NJ, USA: Prentice Hall.

Phillips, F. 2001. *Market Oriented Technology Management: Innovating for Profit in Entrepreneurial Times*. New York, NY, USA: Springer.

Pugh, S. 1990. *Total Design: Integrated Methods for Successful Product Engineering*. Englewood Cliffs, NJ, USA: Prentice Hall.

Reinertsen, D. 1997. *Managing the Design Factory: A Product Developers Tool Kit*. New York, NY, USA: Simon & Schuster Ltd.

Rogers, E.M. 2003. *Diffusion of innovations,* 5th ed. New York, NY: Free Press.

Smith, P., and D. Reinertsen. 1997. *Developing products in half the time – new rules, new tools*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

US Army. 2012. "Chapter 2, section A.4" in *Army Science and Technology Master Plan*. Accessed January 12, 2012. Available at: http://www.fas.org/man/dod-101/army/docs/astmp/c2/P2A4.htm.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Product Systems Engineering Special Activities

Product systems engineering has activities that are unique to products. This article discusses many of them.

Readiness Level Assessments

As a new system is developed, it is essential to verify and validate that the developed system is mature enough to be released as an operational product or service. Technology readiness assessments (TRA) are established tools used to qualify technology development and help make investment decisions within complex development programs in order to deploy systems or elements of technology to an end user in a timely fashion.

This notion of maturity was formalized by the US National Aeronautics and Space Administration (NASA) (Mankins 1995) and later modified for use by the Department of Defense (DoD), the Air Force Research Laboratory (AFRL), and the US Department of Energy (DoE), as well as a growing number of non-governmental organizations. Technology readiness levels (TRL) are a metric developed to summarize the degree of maturity of a technology. The original NASA TRL scale has nine different levels from the *basic principles observed and reported* (TRL 1) to *actual systems "flight proven" through successful mission operations* (TRL 9). The TRL scale utilized by the DoD is portrayed in Table 1.

Table 1. Technology Readiness Levels for Assessing Critical Technologies (Mankins 1995). Released by the Advanced Concept Office, Office of Space Access and Technology, NASA.

Technology Readiness Level	1. Basic principles observed and reported.	2. Technology concept and/or application formulated.	3. An analytical and experimental critical function and/or characteristic proof of concept.	4. Component validation in laboratory environment.	5. Component validation in relevant environment.	6. Prototype demonstration in a relevant environment.	7. Prototype demonstration in an operational environment.	8. System qualified through test and demonstration.	9. System proven through successful mission operations.	Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation.

The utilization of TRLs has an impact on the structure and operation of life cycles as described in Part 3; they allow better management and control of risks inherent with technology, as well as better control of costs and the schedule of program development. However, TRLs do not provide an assessment of the programmatic influence on a TRL, technology criticality and priority, software aging and readiness context, as pointed out by Smith (2005). While TRLs have proven to be useful in evaluating a technology's performance, as demonstrated in the laboratory or in a test environment, they do not inform one whether or not the technology product can actually be produced in an affordable manner. The concept of manufacturing readiness levels (MRL) has been incorporated to expand the TRL idea so that it can incorporate producibility concerns. The MRL approach addresses questions such as the level of technology reproducibility, the cost of production, and technology manufacturing production environment early in the development phase (GAO 2003, DoD 2011).

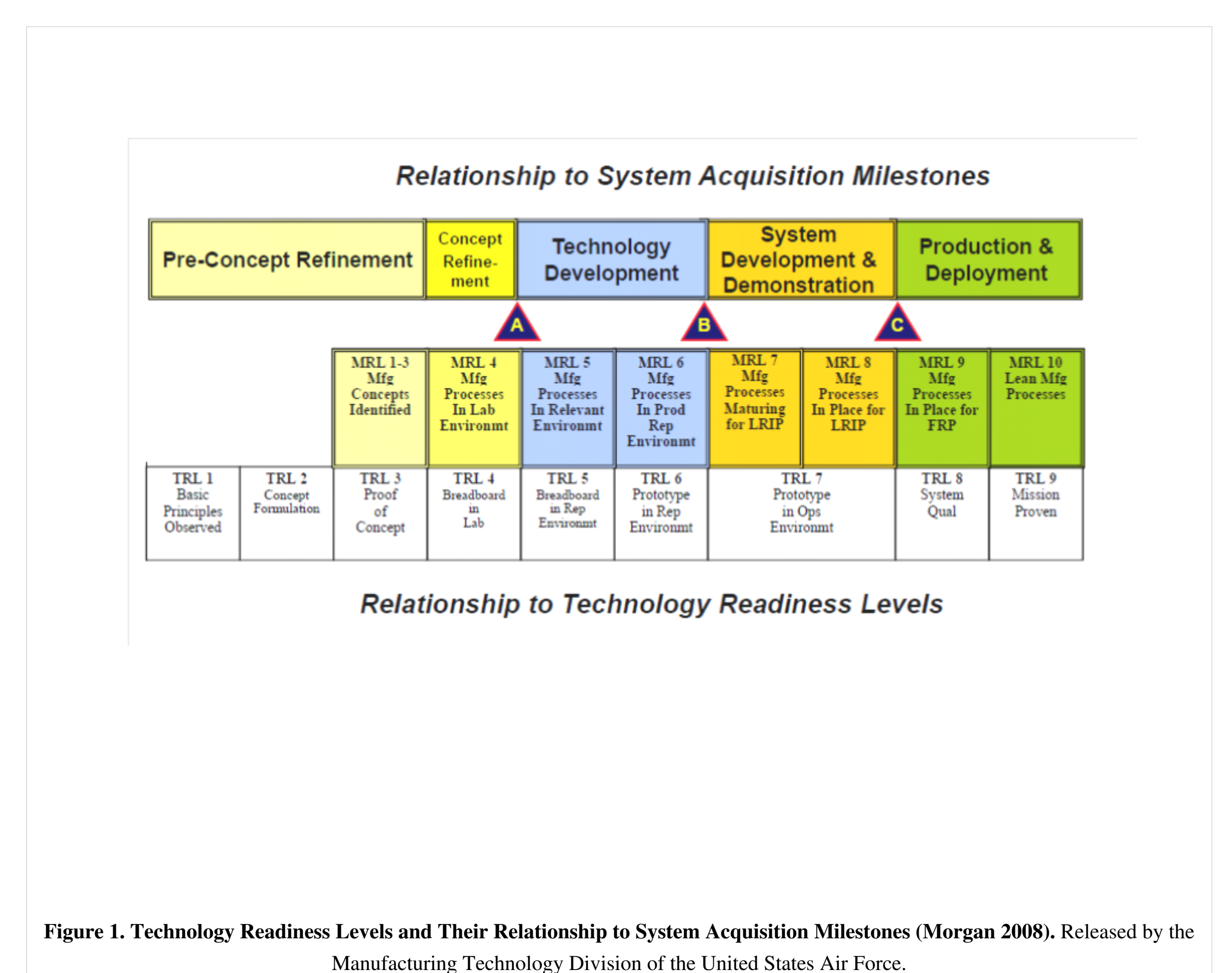


Figure 1. Technology Readiness Levels and Their Relationship to System Acquisition Milestones (Morgan 2008). Released by the Manufacturing Technology Division of the United States Air Force.

Readiness levels are an active research area within academia and government agencies in regards to the integration of technology components into complex systems (integration readiness levels (IRLs)) to address interface maturity among existing and maturing technology developments. TRLs apply to the critical enabling technologies, which are usually embodied at the subsystem, assembly level, or system component level. Systems readiness levels (SRL) are used when going from individual technologies to the whole system. The SRL model is a function of the individual TRLs in a system and their subsequent integration points with other technologies, the IRL (Sauser 2006).

Another maturity aspect is related to the provisioning of products that are readily available and referred to as commercial off-the-shelf (COTS). Such products, be they hardware, software, or a mixture of both, have hopefully achieved the degree of maturity so that those acquiring them can rely upon their operational properties and that the documentation of the COTS products is sufficient to provide the proper guidance in their use.

The PSE should realize that the TRL assessment for COTS changes dramatically if the operational environment or other requirements are imposed that exceed the design limits of the COTS product (e.g., operations at very high or very cold temperatures, high shock, or vibration levels).

Product Certification

Product certifications are both domain and product specific, and typically relate to human safety and health, the need to meet a specific government regulation, or are required by underwriters for insurance purposes. Certifications are performed by a third party (independent of the developer) who provides a guarantee of the quality, safety, and reliability of the product to the customer or user.

The INCOSE *SE Handbook* defines product certification as "the process of certifying that a certain product has passed performance or quality assurance tests or qualification requirements stipulated in regulations such as a building code or nationally accredited test standards, or that it complies with a set of regulations governing quality or minimum performance requirements." (INCOSE 2012)

The INCOSE *SE Handbook* also defines four methods for verification: inspection, analysis, demonstration, and testing (INCOSE 2012). In addition, it defines certification as a fifth verification method, which is defined as verification against legal or industrial standards by an outside authority without direction to that authority as to how the requirements are to be verified. For example, electronic devices require a CE certification in Europe, and a UL certification in the US and Canada (INCOSE 2012).

The best known certification is the airworthiness certification, which relates to the safety of flight for aircraft. In the US, the test for this certification is performed by the Federal Aviation Administration (FAA). Government certifications are also common in the medical systems field where the Federal Drug Administration (FDA) is the primary certification agency. Some certifications are based on standards defined by technical societies, such as the American Society of Mechanical Engineers (ASME). The combination of the technical standards and a certification allows product developers to perform certifications that meet government standards without having the government directly involved in the process.

There are equivalent government organizations in other countries and for other regulated areas, such as communications, building safety, nuclear systems, transportation systems to include ships, trains and automobiles, environmental impact, and energy use. Systems engineers must be aware of the certifications that are required for the domain and product being developed. Certification agencies must be involved early in the development effort to ensure the necessary certifications are included in the system requirements, the system development plan, and the funding provided to accomplish the development. When system changes and upgrades are necessary, the systems engineers must determine if product re-certification is necessary and include it in the plans and funding for the system upgrade.

Enabling Product Certifications

There may be other certifications for enabling products that must be considered and appreciated by PSE, such as an operator certification of airplane pilots to ensure flight safety, and certification of nuclear plant operators to ensure prevention or mitigation of nuclear radiation effects. An example of this is shown in the certification program by the North American Electric Reliability Corporation (NERC):

> *In support of NERC's mission, the System Operator Certification Program's mission is to ensure that employers have a workforce of system operators that meet minimum qualifications. These minimum qualifications are set through internationally recognized processes and procedures for agencies that certify persons. The Certification Program promotes excellence in the area of system operator performance and encourages system operators to be inquisitive and informed.* (NERC 2012)

Production qualification testing (PQT) is another type of certification which DAU (2005) describes as:

A technical test completed prior to the full-rate production (FRP) decision to ensure the effectiveness of the manufacturing process, equipment, and procedures. This testing also serves the purpose of providing data for the independent evaluation required for materiel release so that the evaluator can address the adequacy of the materiel with respect to the stated requirements. These tests are conducted on a number of samples taken at random from the first production lot, and are repeated if the process or design is changed significantly and when a second or alternative source is brought online.

Security certification and accreditation (C&A) is often required for the deployment of computing and networking equipment in a classified environment. Facility certification may be required to ensure that a building housing the equipment can provide the proper environment for safe and efficient operation of the equipment. High-altitude electromagnetic pulse (HEMP) certification may be required to ensure that a building and its equipment can withstand the effects of HEMP from nuclear weapons. A similar type of certification to HEMP is TEMPEST testing to ensure that sensitive electronic emissions are not allowed to leave high security facilities. TEMPEST is a code name referring to investigations and studies of compromising emission, and is not an acronym.

Technology Planning and Insertion

Technology planning can be an enterprise function or a program function. Technology planning as an enterprise function typically occurs on an annual basis to determine the funding necessary for independent research and development in the coming year. Technology planning as a program function occurs early in the program and often continues throughout the life of the system. The design of the product system is highly dependent on the availability of technologies that have acceptable risks and that meet the customer's cost, schedule, and performance requirements. These critical technologies will only be available when necessary if the systems engineers perform concept designs, technology assessments, and trade studies that define the critical technologies and the capabilities necessary before the system development activities that will use the critical technologies begin.

The MITRE *Systems Engineering Guide* (MITRE 2011) provides the following definition for technology planning:

Technology Planning is the process of planning the technical evolution of a program or system to achieve its future vision or end-state. Technology planning may include desired customer outcomes, technology forecasting and schedule projections, technology maturation requirements and planning, and technology insertion points. The goal is a defined technical end-state enabled by technology insertion over time.

Systems engineers who participate in technical planning must understand the future vision and system requirements, and relate these to the current and expected future technologies that can be applied to the system design during current development stages, as well as for potential future upgrades to the system. To do this, systems engineers must acquire and maintain knowledge of the existing and developing technology in their design domain. The systems engineer will also provide the essential connection between the system user and research communities to provide alignment between the technology developers and the system designers.

Technology planning and insertion usually requires that the systems engineer perform technology readiness assessments that rate the maturity levels and the risks associated with the planned technologies. Immature, risky technologies require risk reduction activities that include prototyping and product development and test activities that provide quantification of the capabilities and risks. The risk reduction activities provide the data necessary to assess and update the design to reduce its risk.

Product Road Mapping and Release Planning

Product road maps provide an outline that shows when products are scheduled for release and include an overview of the product's primary and secondary features. Both internal and external product road maps should be created. The form of the road map will depend on the development methodology being used. Waterfall, iterative, and spiral development models result in different road maps and release plans. The systems engineer must be an integral member of the team that creates road maps. Requirements should be mapped onto each of the planned releases. Test plans must be adapted to the development model and the release plans.

Product road maps should be aligned with the technology road maps that are applicable to the product. Technology maturity should be accomplished before the technologies are included in the product development plans and the road map for the product release that includes those technologies.

Product road maps are essential for software intensive systems that have many releases of software and capability upgrades. The identification of the requirements, the test plans, and the features provided for each release are an essential driver of the product development process. Clear definition of these items can make the difference between delivering the capabilities the customer is looking for and will support, or a product that fails to meet the needs of the customer and is abandoned.

Intellectual Property Management

Systems engineers must also manage intellectual property as part of their job. Existing systems engineering literature rarely covers this topic. However, there are many textbooks and management related literature that provide additional information, such as “Intellectual Property Rights for Engineers” (Irish 2005). Intellectual property may be considered as *intangible output of the rational thought process that has some intellectual or informational value and is normally protected via using copyrights, patents, and/or trade secrets* (Irish 2005). Listed below are some of the more important intellectual property types with brief explanations:

- Proprietary Information: Any information which gives a company (or enterprise) an advantage over its competitors is usually proprietary.
- Patents: A patent is the principle mechanism for protecting rights for an invention or discovery. In exchange for a full disclosure of how to practice it, the issuing government will grant the right to exclude others from practicing the invention for a limited amount of time, usually 15 to 20 years (in the US, a patent usually lasts for 17 years from the date of issue).
- Design Patents: In some countries, these are referred to by the more appropriate term *design registrations* or some other name. They protect rights in ornamental designs, provided the designs are new and inventive, i.e., *non-obvious* at the time they are made. In the US, the maximum length of a design patent is 14 years.
- Trademarks: A trademark identifies the source of origin for goods in commerce, and is not stronger than the actual use to which it has been put to and the diligence with which it has been protected from infringement, encroachment, or dilution. Under some circumstances, a trademark may be registered with governmental agencies. Among a company's most valuable assets is the corporate name, which also is the company's primary trademark.
- Copyrights: A claim of copyright protects such works as writings, musical compositions, and works of art from being copied by others, i.e., from plagiarism. A notice of claim of copyright must be made in the manner prescribed by law at the time of a protected work’s first publication.

Parts, Materials, and Process Management

The consequences of mission failure or an inability to deploy the system on time due to parts, materials, and process (PM&P) issues needs to be clearly understood by the systems engineer since these elements are fundamental to the overall mission reliability and program success. PM&P management is especially important in harsh environments (like outer space and underwater) and in situations where system failure can have catastrophic impacts on public safety (like nuclear power, bridges and tunnels, and chemical processing plants).

Generally, original equipment manufacturers (OEMs) engaged in the design and fabrication of electronic systems have a documented policy that deals with PM&P, sometimes in the form of a PM&P Management Manual. The elements of a PM&P control program include things such as

- PM&P requirements that apply to a system;
- the generation number of a program or project approved parts list (PAPL);
- the appointment of a PM&P control board (PMPCB);
- the development of a part stress derating policy and a part parameter derating policy for end of life use; and
- a definition of the minimum qualifications, quality controls, and screening requirements for parts.

PM&P management guidance is provided by MIL-HDBK-512 (DoD 2001) and ANSI/AIAA R-100 (2001), which identify the overall management process elements of a PM&P program. Additional issues to be addressed by PM&P include the following: hazardous materials, rare earth elements, conflict materials, and counterfeit materials.

References

Works Cited

ANSI/AIAA. 2001. *Recommended Practice for Parts Management*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/American Institute of Aeronautics and Astronautics (AIAA), ANSI/AIAA R-100A-2001.

DAU. 2005. *Glossary of Defense Acquisition Acronyms & Terms*, 12th ed. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense (DoD). Available at: http://www.dau.mil/pubscats/PubsCats/13th_Edition_Glossary.pdf.

DoD. 2000. *Department of Defense Directive (DoD-D) 5000.01: The Defense Acquisition System*. Arlington, VA, USA: US Department of Defense, Acquisition, Technology, and Logistics (AT&L). Available at: http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf.

DoD. 2001. *Department of Defense Handbook: Parts Management*. Arlington, VA, USA: Department of Defense (DoD). MIL-HDBK-512A.

DoD. 2011 *Department of Defense Technology Readiness Assessment (TRA) Guidance*, Assistant Secretary of Defense for Research and Engineering (ASD(R&E)), May 2011.

FAA. 2011. *Airworthiness Certificates Overview.* Washington, DC, USA: Federal Aviation Administration (FAA). Available at: http://www.faa.gov/aircraft/air_cert/airworthiness_certification/aw_overview/.

GAO. 2003. *Defense acquisitions: Assessments of Major Weapon Programs*, GAO-03-476, US Government Accountability Office, May 2003.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Irish, V. 2005. *Intellectual Property Rights for Engineers*, 2nd ed. Herts, UK: Institution of Engineering and Technology (IET).

Mankins, J. 1995. *Technology Readiness Levels—A White Paper*. Washington, DC, USA: Advanced Concepts Office, Office of Space Access and Technology, National Aeronautics and Space Administration (NASA).

MITRE. 2011. "Systems Engineering Guide." Accessed September 11, 2012. Available at: http://www.mitre.org/work/systems_engineering/guide/.

Morgan, J 2007. *Manufacturing Readiness Levels (MRLs) and Manufacturing Readiness Assessments (MRAs)*. ADA510027 Air Force Research Lab Wright-patterson Afb Oh Manufacturing Technology Directorate. September 2007. Accessed 06 November 2014 at Defense Technical Information Center http://www.dtic.mil/get-tr-doc/pdf?AD=ADA510027

NERC. 2012. "North American Electric Reliability Corporation (NERC)." Accessed September 11, 2012. Available at: http://www.nerc.com.

Sauser, B., D. Verma, J. Ramirez-Marquez, and R. Gove. 2006. *From TRL to SRL: The Concept of System Readiness Levels*. Proceedings of the Conference on Systems Engineering Research (CSER), April 7-8, 2006, Los Angeles, CA, USA.

Smith, J. 2005. *An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software*. Proceedings of the 38th Hawaii International Conference on Systems Sciences, January 3-6, 2005, Island of Hawaii, USA.

Primary References

Mankins, J. 1995. *Technology Readiness Levels—A White Paper*. Washington, DC, USA: Advanced Concepts Office, Office of Space Access and Technology, National Aeronautics and Space Administration (NASA).

MITRE. "Systems Engineering Guide." Available at http://www.mitre.org/work/systems_engineering/guide/

Sauser, B., D. Verma, J. Ramirez-Marquez, and R. Gove. 2006. *From TRL to SRL: The Concept of System Readiness Levels*. Proceedings of the Conference on Systems Engineering Research (CSER), Los Angeles, CA, April 7-8, 2006.

Additional References

ANSI/AIAA. 2001. *Recommended Practice for Parts Management*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/American Institute of Aeronautics and Astronautics (AIAA), ANSI/AIAA R-100A-2001.

DoD. 2000. *Department of Defense Directive (DoD-D) 5000.01: The Defense Acquisition System*. Arlington, VA, USA: US Department of Defense, Acquisition, Technology, and Logistics (AT&L). Available at: http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf.

DoD. 2001. *Department of Defense Handbook: Parts Management*. Arlington, VA, USA: Department of Defense (DoD). MIL-HDBK-512A.

FAA. 2011. "Airworthiness Certificates Overview." Washington, DC, USA: Federal Aviation Administration (FAA). Available at: http://www.faa.gov/aircraft/air_cert/airworthiness_certification/aw_overview/.

Irish, V. 2005. *Intellectual Property Rights for Engineers*, 2nd ed. Herts, UK: Institution of Engineering and Technology (IET).

Smith, J. 2005. *An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software*. Proceedings of the 38th Hawaii International Conference on Systems Sciences, January 3-6, 2005, Island of Hawaii, USA.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Service Systems Engineering

Service Systems Engineering

The growth of services in the ever-evolving global economy has brought much needed attention to service science and service systems engineering (SSE). Research focuses on developing formal methodologies to understand enterprise-end-user (customer) interactions from both socio-economic and technological perspectives, and to enable value co-creation and productivity improvements. Service systems require trans-disciplinary collaborations between society, science, enterprises, and engineering. Service transactions are customized and personalized to meet a particular customer need. This requires a disciplined and systemic approach among stakeholders and resources to emphasize end-user satisfaction in the design and delivery of the service (Hipel et al. 2007; Tien and Berg 2003; Vargo and Akaka 2009; Maglio and Spohrer 2008; Maglio et al. 2010).

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Service Systems Background
- Fundamentals of Services
- Properties of Services
- Scope of Service Systems Engineering
- Value of Service Systems Engineering
- Service Systems Engineering Stages

Introduction

New Service Development (NSD) has usually been a proprietary process closely guarded by product businesses and service businesses for their competitive advantage. Traditional systems engineering practices have been primarily applied in aerospace and defense sectors while SSE practices have been applied by information and communications technologies (ICT) service providers (Booz, Allen, and Hamilton 1982; Johnson et al. 2000; Eppinger 2001; Freeman 2004; Whitten and Bentley 2007; AT&T SRP 2008; Lin and Hsieh 2011).

These early efforts were, and in some instances remain, very important for product and service businesses. However, the growth and ubiquity of the World Wide Web, advances in computer science and ICT, and business process management through "social networking," support the realization of closely interrelated service systems. Product business (manufacturing, agriculture, etc.) and service business distinctions are going away (Spohrer 2011).

These services, or service innovations, must take into account social aspects, governance processes, business processes, operational processes, as well as design and development processes. The customer, service provider, product provider, and intermediaries need to collaborate toward the optimization of customer experiences and customer provided value (through co-creation). The interrelations among different stakeholders and resources require that methodologies, processes, and tools be dynamically tailored and delivered for either foreseen or newly discovered services to rapidly adapt to changing enterprise and end-user environments.

Even in the case of static, predetermined, interaction rules, the major problems faced in the definition, design, and implementation of services have been in understanding the integration needs among different systems, system entities, stakeholders, and in defining the information flows required for the governance, operations, administration, management and provisioning (OAM&P) of the service. (Maier 1998; Jamshidi 2008; Pineda 2010; Luzeaux and Ruault 2013). Thus, the 21st century technology-intensive services are “information-driven, customer centric, e-oriented, and productivity-focused" as discussed by Chesbrough (2011), Chang (2010), Tien and Berg (2003), and Hipel et al. (2007). A detailed discussion of these characteristics is given in the Value of Service Systems Engineering article within this KA.

Service Systems Engineering Knowledge Area Topics

This knowledge area (KA) describes best practices in SSE during the service design process and outlines current research on methods, processes, and tools. It does not attempt to describe the initial efforts and research in service science that were proposed and introduced by International Business Machines (IBM) (Maglio and Spohrer 2008), but it does recognize their leadership in championing these concepts in undergraduate and graduate curricula.

The rest of the KA is organized in the following way:

The Service Systems Background article presents some background on the transition from a manufacturing economy toward the service economy brought by the World Wide Web through co-creation of end-user value. It describes how this transformation is impacting industries, such as healthcare, agriculture, manufacturing, transportation, supply chain, environmental, etc. The article also describes the scope of the SSE discipline's contributions to meeting the needs of the service sector companies in strategic differentiation and operational excellence (Chang 2010) by pointing out some differences between product-oriented systems engineering and SSE.

The Fundamentals of Services and Properties of Services articles take the reader through a general discussion of services and current attempts to classify different types of services, in particular, attention is paid to the properties of service systems for the service sector, such us transportation, environmental and energy services, consulting services, healthcare, etc.

The Scope of Service Systems Engineering and Value of Service Systems Engineering articles cover the value of SSE, defining (or using when available) service architecture frameworks, and the stages of the service development process from concept to life cycle management.

The Service Systems Engineering Stages article summarizes the major SSE process activities that need to be carried out during the service design process and the needed output (work products) in each of the service design process stages.

Service Innovation and Value-Co-creation

Service innovation has several dimensions. Service innovation can come about through the creation of a service concept which is sufficiently different that it is not merely an improved service, but in reality is a new service concept. To maintain the rigor and value of innovation, it is necessary to distinguish between an improved service, which may generate some additional value, and a truly new and innovative service concept, which may generate a great deal of value. Dr. Noriaki Kano, a renowned quality management guru, has suggested that every service concept has its inherent attributes and we should strive to continuously improve upon these; but this is not innovation (Kano 1996).

To be innovative, the change in a value proposition cannot be incremental, but it must be enough to significantly impact customer and competitor behavior (e.g., new market creation). Value innovation involves a shift in perspective of customer needs that requires a rethinking of what service value proposition is delivered (Kano 1996).

Innovation can also come through a significant change in the way or the reason the customer is engaged or connected. In a service value chain the customer may well change from being just a receiver of service value to

becoming a co-creator, or an active participant in the design and delivery, i.e., service transaction of service value. At the retail level, when a customer designs the time, route, and price selection for a plane ticket purchased online, he is co-creating the service. Value innovation involves a shift in perspective of customer needs that requires a rethinking of how a service value proposition is delivered (Bettencourt 2010).

Finally, service innovation can come through significant changes in the way the enterprise is organized to create a service value proposition from concept through delivery. A considerable improvement in the enterprise structure and/or governance can be seen as innovation. Value innovation involves a shift in perspective of customer needs that requires a rethinking of how an enterprise organizes to support a service value proposition.

Continuous improvement can be reasonably planned and predicted while innovation and breakthroughs cannot. The most effective way to obtain innovation and breakthroughs is to encourage the culture, environment, and atmosphere that are conducive to innovation and breakthroughs. Innovative co-creation requires the integration of people, ideas, and technology for the purpose of creating value for themselves, their customers, companies, and society.

The lone inventor sees a problem and must work to create the solutions to all dimensions of the problem. Co-creators see the problem and realize that there may already be several creators, each already having a piece of the solution. Co-creation embraces the value of things “not invented here” because of the velocity they can bring to ideation and time to market. This service innovation process is facilitated by modern mass (and at the same time, personal) communication technology evident in social networking platforms.

Towards a Discipline of Service Systems Engineering

Mindful of the evolution taking place in the global economy and the world markets, it would be futile to attempt covering all the major advances and the boundless possibilities in the services sector for the rest of the century. The services sector covers wide areas of application studied in many different fields (e.g., business science, social science, cognitive science, political science, etc.). The field of service systems, a trans-disciplinary analysis and study of services, was only introduced 10 to 15 years ago. As a consequence, much of the existing literature on services and service-innovation is scattered. The main objective of this KA is to document the systems engineering processes, methodologies, and existing tools as applied to the service design process, and to introduce critical SSE challenges and research areas.

References

Works Cited

AT&T SRP. 2008. *Technical Approach to Service Delivery*. General Services Administration, AT&T Bridge Contract No. GS00Q09NSD0003. Accessed on June 1, 2011. Available at: http://www.corp.att.com/gov/contracts/fts_bridge/technical/07_vol_I_section_1.pdf.

Bettencourt, L. 2010. *Service Innovation: How to Go from Customer Needs to Breakthrough Services*. New York, McGraw-Hill Professional. July 2010.

Booz, Allen, and Hamilton. 1982. *New Products Management for the 1980s*. New York, NY, USA: Booz, Allen, and Hamilton Inc.

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence*. New York, NY, USA: John Wiley & Sons, Inc.

Chesbrough, H. 2011. *Open Services Innovation: Rethinking Your Business to Grow and Compete in a New Era*. San Francisco, CA, USA: Jossey-Bass.

Eppinger, S. 2001. "Innovation at the Speed of Information" *Harvard Business Review*. 79 (1): 149-158.

Freeman, R.L. 2004. *Telecommunication Systems Engineering*, 4th ed. New York, NY, USA: John Wiley & Sons.

Hipel, K.W., M.M. Jamshidi, J.M. Tien, and C.C. White. 2007. *"The Future of Systems, Man, and Cybernetics: Application Domains and research Methods*. IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews. 37 (5): 726-743.

Jamshidi M, *System of Systems Engineering: Innovations for the Twenty-First Century*. New York, NY, USA: John Wiley & Sons. November 2008.

Johnson, S.P., L.J. Menor, A.V. Roth, and R.B. Chase. 2000. "A critical evaluation of the new service development process: integrating service innovation and service design," in Fitzsimmons, J.A., and M.J. Fitzsimmons (eds.). *New Service Development - Creating Memorable Experiences*. Thousand Oaks, CA, USA: Sage Publications. p. 1-32.

Kano, N. 1996. *Guide to TQM in Service Industry*. Tokyo, Japan: Asian Productivity Organization.

Lin, F.R., and P.S Hsieh. 2011. A SAT View on New Service Development." *Service Science*. 3 (2): 141-157.

Luzeaux, D. and Ruault. J,. *System of Systems*. New York, NY, USA: John Wiley & Sons. March 2013.

Maglio, P., C. Kieliszewski, and J. Spohrer. 2010. *Handbook of Service Science*, 1st ed. New York, NY, USA: Springer Science + Business Media.

Maglio, P., and J. Spohrer. 2008. "Fundamentals of Service Science." *Journal of the Academy of Marketing Science*. 36 (1): 18-20.

Maier, M.W., 1998. "Architecting Principles for System of Systems." *Systems Engineering*. 1 (4): 267-284.

Pineda, R. 2010. "Understanding Complex Systems of Systems Engineering." Presented at Fourth General Assembly, Cartagena Network of Engineering, September 21-24, 2010, Metz, France.

Spohrer, J.C. 2011. "Service Science: Progress & Directions." Presented at International Joint Conference on Service Science, May 25-27, 2011, Taipei, Taiwan.

Tien, J.M., and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering*. 12 (1): 13-38.

Vargo, S.L., and R.F. Lusch. 2004. "The Four Service Marketing Myths – Remnants of a Goods-Based Manufacturing Model." *Journal of Service Research*. 6 (4): 324-335.

Vargo, S.L., and M.A. Akaka. 2009. "Service-Dominant Logic as a Foundation for Service Science: Clarifications." *Service Science*. 1 (1): 32-41.

Whitten, J., and L. Bentley. 2007. *Systems Analysis and Design Methods*. New York, NY, USA: McGraw-Hill Higher Education.

Primary References

Maglio, P., C. Kieliszewski, and J. Spohrer. 2010. *Handbook of Service Science*, 1st ed. New York, NY, USA: Springer Science + Business Media.

Tien, J.M., and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering*. 12 (1): 13-38.

Vargo, S.L., and R.F. Lusch. 2004. "The Four Service Marketing Myths – Remnants of a Goods-Based Manufacturing Model." *Journal of Service Research*. 6 (4): 324-335.

Additional References

AT&T SRP. 2008. *Technical Approach to Service Delivery*. General Services Administration, AT&T Bridge Contract No. GS00Q09NSD0003. Accessed on June 1, 2011. Available at: http://www.corp.att.com/gov/contracts/fts_bridge/technical/07_vol_I_section_1.pdf.

Booz, Allen, and Hamilton. 1982. *New Products Management for the 1980s*. New York, NY, USA: Booz, Allen, and Hamilton Inc.

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence*. New York, NY, USA: John Wiley & Sons, Inc.

Eppinger, S. 2001. "Innovation at the Speed of Information." *Harvard Business Review*. 79 (1): 149-158.

Freeman, R.L. 2004. *Telecommunication Systems Engineering*, 4th ed. New York, NY, USA: John Wiley & Sons.

Hipel, K.W., M.M. Jamshidi, J.M. Tien, and C.C. White. 2007. "The Future of Systems, Man, and Cybernetics: Application Domains and research Methods." *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*. 37 (5): 726-743.

Johnson, S.P., L.J Menor, A.V. Roth, and R.B. Chase. 2000. "A critical evaluation of the new service development process: integrating service innovation and service design," in Fitzsimmons, J.A., and M.J. Fitzsimmons (eds.). *New Service Development - Creating Memorable Experiences*. Thousand Oaks, CA, USA: Sage Publications. p. 1-32.

Kano, N. 1996. *Guide to TQM in Service Industry*. Tokyo, Japan: Asian Productivity Organization.

Lin, F.R., and P.S Hsieh. 2011. A SAT View on New Service Development." *Service Science*. 3 (2): 141-157.

Luzeaux, D. and Ruault, J. 2013. *Systems of Systems*, Wiley.

Maier, M.W. 1998. "Architecting Principles for System of Systems." *Systems Engineering*. 1 (4): 267-284.

Maglio, P., and J. Spohrer. 2008. "Fundamentals of Service Science." *Journal of the Academy of Marketing Science*. 36 (1): 18-20.

Pineda, R. 2010. "Understanding Complex Systems of Systems Engineering." Presented at Fourth General Assembly, Cartagena Network of Engineering, September 21-24, 2010, Metz, France.

Spohrer, J.C. 2011. "Service Science: Progress & Directions." Presented at International Joint Conference on Service Science, May 25-27, 2011, Taipei, Taiwan.

Vargo, S.L., and M.A. Akaka. 2009. "Service-Dominant Logic as a Foundation for Service Science: Clarifications." *Service Science*. 1 (1): 32-41.

Whitten, J., and L. Bentley. 2007. *Systems Analysis and Design Methods*. New York, NY, USA: McGraw-Hill Higher Education.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Service Systems Background

Economies are pre-disposed to follow a developmental progression that moves them from heavy proportional reliance on agriculture and mining toward the development of manufacturing, and finally toward more service-based economic activity. As reported by the Organization for Economic Co-Operation and Development (OECD) in its “Science, Technology, and Industry (STI) Forum on The Service Economy":

> *The reason that we see a services economy today, and gather to talk about it and recognize its importance is because technology has allowed service industries to gain the operational leverage that manufacturing achieved 100 years ago. In addition to banks, health systems, telephone and telecommunications networks, and distribution and retailing firms are further examples of sectors that have been able to benefit from economies of scale. As a result, we are now living in a world where global-scale service companies exist for the first time, whereas we have seen global manufacturing companies for 50 years or more.* (OECD 2000, 8)

Evolution Toward Service-Based Economies

The typical industry example given of this progression toward services is the company International Business Machines (IBM). Even though IBM still produces hardware, they view their business as overwhelmingly service-oriented wherein hardware plays only an incidental role in their business solutions services; the fastest line of business growth within IBM has been the business-to-business (B2B) services: information technology (IT); for example, data centers and call centers; business process outsourcing/re-engineering; systems integration; and organizational change.

Business to government (B2G) is forecasted to have the fastest growth in the years to come (Spohrer 2011). For IBM, this trend started in 1989 with the launch of business recovery services; it accelerated with the acquisition of Price-Waterhouse Coopers Consultants in 2002 and culminated with the 2005 sale of the laptop (ThinkPad) manufacturing, their last major hardware operation.

IBM exemplifies the services trend which has accelerated in the last 25-30 years and as of 2006, the services produced by private industry accounted for 67.8% of U.S. gross domestic product (GDP). The top sub-sectors included real estate, financial, healthcare, education, legal, banking, insurance, and investment. Production of goods accounted for 19.8% of GDP. The top product sub-sectors included manufacturing, construction, oil and gas, mining, and agriculture (Moran 2006).

Beginning in the mid-1990s, the concept of a product-service system (PSS) started to evolve. PSSs have been adopted by businesses interested in using the model to bring not only added value to their existing offerings, but capital-intensive, environmentally favorable products to market (Mont and Tukker 2006).

There are some definitional issues in any discussion of PSS, including the fact that services can sometimes be considered as products, and services invariably need physical products to support their provisioning or delivery (2006). A PSS is comprised of tangibles and intangibles (activities) in combination to fulfill specific customer requirements, or ideally, to allow applications to be co-created flexibly by linking loosely coupled agents, typically over a network (Domingue et al. 2009). Research has shown that manufacturing firms are more amenable to producing "results" rather than solely products as specific artifacts and that end users are more amenable to consuming such results (Cook 2004; Wild et al. 2007).

The popularity of wikis, blogs, and social networking tools is strong evidence that "Enterprise 2.0" is already well under way; Andrew McAfee describes Enterprise 2.0 as "the use of emergent social software platforms within companies, or between companies and their partners or customers" (McAfee 2009). However, the integrated access to people, media, services, and things, provided by the Future Internet, will enable new styles of societal and economic interactions at unprecedented scales, flexibility, and quality. These applications will exploit the wisdom of

crowds and allow for mass collaboration and value co-creation.

The future internet will provide location independent, interoperable, scalable, secure, and efficient access to a coordinated set of services (Tselentis et al. 2009), but such a broad vision demands a sound and well-defined approach for management and governance.

Current application service providers like Amazon, Facebook, Twitter, eBay, and Google must mediate between the business challenges enabled by network and IT convergence and customers (enterprise or consumer) demanding new and more value-adding services enabled by social networks (TMFORUM 2008). The differences between IT and communications technologies are disappearing; internally-focused processes (back-stage processes) for operations optimization are now being strongly tied to the customer facing (front-stage) processes for value co-creation and delivery. In this scenario, the enterprise's internal organization and employees are embedded in the service value chain to benefit customers and stakeholders. In the service-dominant logic (S-DL) for marketing (Vargo and Lusch 2004), service is the application (through deeds, processes, and performances) of specialized operant resources (knowledge and skills) for the benefit of another entity or the entity itself. The emphasis is on the process of doing something for, and with, another entity in order to create value; a service system is thus a system of interacting and interdependent parts (people, technologies, and organizations) that is externally oriented to achieve and maintain a sustainable competitive advantage (IFM 2008; Maglio and Spohrer 2008).

The future internet is expected to be more agile, scalable, secure, and reliable, demanding rapidly emerging applications/services with different requirements and implications for the Future Internet design that pose a significant set of problems and challenges, in particular, "the fragmentation of knowledge and the isolation of the specialist as well as the need to find new approaches to problems created by earlier 'solution of problems,'" (Skyttner 2006). The service systems engineering discipline may inform the discussion and offer potential multidisciplinary environments and trans-disciplinary solutions.

The internet has been successfully deployed for several decades due to its high flexibility in running over different kinds of physical media and in supporting different high-layer protocols and applications, including traditional file transfer, email, and client-server-based Web applications, among others.

Business Dependence on Service Systems

Most people and enterprises are heavily dependent on service interactions, including entertainment, communications, retail, education, healthcare, etc., brought about by emerging services, such as video on demand, web conferencing, time-shift services, place-shift and device-shift services, enterprise applications (e.g., enterprise resource planning (ERP), customer relationship management (CRM), manufacturing resource management (MRM), software configuration management (SCM), etc.), software as a service (SaaS), platform as a service (PaaS), cloud services, peer-to-peer (P2P) services, etc. A common denominator in the set of services mentioned is that applications are offered as services by the interaction of service system entities and thus they are service based applications (SBA).

Thus, "A service based application is obtained by composing various service system entities to satisfy the desired functionality" (Andrikopoulos et al. 2010). SBAs are heavily dependent on web services development, such as Web services 2.0 (WS). Software systems engineering (SwSE) plays a very important role in a business dependent on a service system. However, another important role is played by human interfaces, organizational development and technology development; for instance, governance (rules & regulations) and technology research and development are required for future services in healthcare services, intelligent transportation services, environmental services, energy services, etc. to address societal challenges of the 21st century (sustainability, energy, etc.) as presented by (Vest 2010) if we were to face those challenges as an ecosystem.

Service System Example

In an intelligent transport system-emergency transportation operation (ITS-ETO), the service goal is to provide safe evacuation, prompt medical care, and improved emergency management service. Typically, a traveler can request service through an emergency call or automated crash report feature, or a public safety officer on location can request service based on customer features and access rights.

The ITS-ETO service system utilizes advances in communication and information systems (technology and information enabler) to access essential, real-time data about conditions on routes throughout the affected area and coordinate operational and logistical strategies in cooperation within all service entities (organization processes). In a critical emergency situation, when patient conditions are continuously changing, ITS can help identify the appropriate response and get the correct equipment (infrastructure enabler), such as a helicopter and emergency personnel (people enabler), to and from the scene quickly and safely.

Efficient and reliable voice, data, and video communications (application enabler) further provide agencies with the ability to share information related to the status of the emergency, the operational conditions of the transportation facilities, and the location of emergency response resources to help communicate and coordinate operations and resources in real time. Advances in logistical and decision-making tools can enable commanders and dispatchers to implement strategies as conditions change (decision making).

It is also critical to receive information on the environmental conditions (storm, hazardous materials, multi-vehicle crashes, etc.) and/or road closures when coordinating evacuations. The availability of real-time data about transportation conditions, coupled with decision-making tools, enables more effective responses and coordination of resources during emergencies. ITS-ETO also enhances the ability of transportation agencies to coordinate responses with other stakeholders/entities.

As a result, increased data accuracy, timeliness, and automation leads to better use of resources, and reuse of exchanges, resulting in time and cost savings. Enhanced response and management leads to greater situational awareness and more effective reactions with the ability to identify and utilize the appropriate equipment, resulting in a more efficient response at the right time (output) (US DOT 2011). Figure 1 below lists the possible stakeholders in a service system.

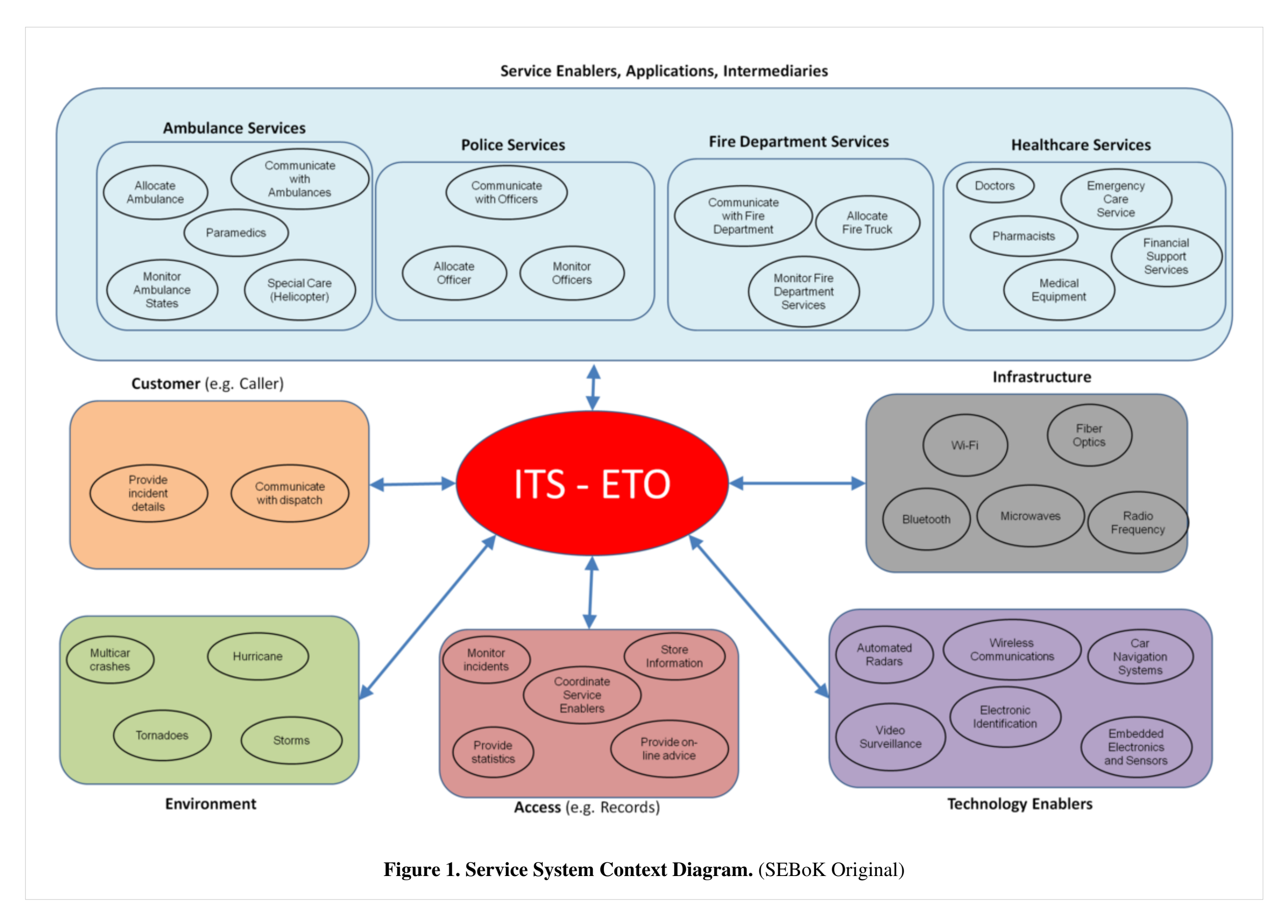


Figure 1. Service System Context Diagram. (SEBoK Original)

As seen in the above example, the service activities are knowledge-intensive; well defined linkages (including access rights) and relationships among different entities give rise to the needed service systems interactions for the service system to be successful. As the world becomes more widely interconnected, and people become better educated, the services networks created by the interaction of the service systems will be accessible from anywhere, at any time, by anyone with the proper access rights.

Knowledge agents are then humans creating new linkages of information to create new knowledge which “can later be embedded in other people, technology, shared information, and organizations." Thus, people can be considered as individual service systems with “finite life cycles, identities (with associated histories and expectations), legal rights and authority to perform certain functions, perform multitasking as a way to increase individual productivity output in a finite time, and engage in division-of-labor with others to increase collective productive output in finite time” through service transactions enabled by their access rights (Spohrer and Kwan 2008).

References

Works Cited

Andrikopoulos, V., A. Bucchiarone, E. Di Nitto, R. Kazhamiakin, S. Lane, V. Mazza, and I. Richardson. 2010. "Chapter 8: Service Engineering," in *Service Research Challenges and Solutions for the Future Internet S-Cube – Towards Engineering, Managing and Adapting Service-Based Systems*, edited by M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger. Berlin Heidelberg, Germany: Springer-Verlag. p. 271-337.

Cook, M. 2004. "Understanding The Potential Opportunities Provided by Service-Orientated Concepts to Improve Resource Productivity," in *Design and Manufacture for Sustainable Development 2004*, edited by T. Bhamra and B. Hon. Bury St. Edmonds, Suffolk, UK: Professional Engineering Publishing Limited. p. 123-134.

Domingue, J., D. Fensel, J. Davies, R. González-Cabero, and C. Pedrinaci. 2009. "The Service Web: a Web of Billions of Services," in *Towards the Future Internet - A European Research Perspective*, edited by G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zehariadis. Amsterdam, The Netherlands: IOS Press.

IFM. 2008. *Succeeding through Service Innovation: A service perspective for education, research, business and government*. University of Cambridge Institute for Manufacturing (IfM) and International Business Machines Corporation (IBM) report. Cambridge Service Science, Management and Engineering Symposium, July 14-15, 2007, Cambridge, UK.

Maglio P., and J. Spohrer 2008. "Fundamentals of Service Science." *Journal of the Academy of Marketing Science*. 36 (1): 18-20. DOI: 10.1007/s11747-007-0058-9.

Maglio, P., Weske, M., Yang, J. and Fantinato, Marcelo. 2010. *8th International Conference on Service Oriented Computing (ICSOC 2010)*. Lecture Notes in Computer Science. Vol. 6470. Springer-Verlag, San Francisco, California. December 2010.

McAfee, A. 2009. *Enterprise 2.0: New Collaborative Tools for Your Organization's Toughest Challenges*. Boston, MA, USA: Harvard Business School Press.

Mont, O., and A. Tukker. 2006. "Product-Service Systems." *Journal of Cleaner Production*. 14 (17): 1451-1454.

Moran, M. 2006. *Servicizing Solar Panels.* Industry Course Report. Lund University International Master's Programme in Environmental Studies and Sustainability Science Department (LUMES), Lund University, Sweden.

Organization for Economic Co-operation and Development (OECD). 2000. *The Service Economy*. Science Technology Industry (STI) Business and Industry Policy Forum Series. Paris, France: OECD. Available: http:// www.oecd.org/dataoecd/10/33/2090561.pdf.

Skyttner, L. 2006. *General Systems Theory: Perspectives, Problems, Practice*, 2nd ed. Singapore: World Scientific Publishing Company.

Spohrer, J., and S.K. Kwan. 2009. "Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline - Outline & References." *International Journal of Information Systems in the Service Sector*. 1 (3): 1-31.

Spohrer, J.C. 2011. "Service Science: Progress & Directions." Presented at the International Joint Conference on Service Science, May 25-27, 2011, Taipei, Taiwan.

TM Forum. 2008. *Service Delivery Framework (SDF) Overview, Release 2.0*. Morristown, NJ: TeleManagement Forum. Technical Report 139.

Tselentis, G., J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zahariadis (eds.). 2009. *Towards the Future Internet - A European Research Perspective*. Amsterdam, The Netherlands: IOS Press.

US DOT. 2011. "Emergency Transportation Operations." Research and Innovative Technology Administration. Accessed June 23, 2011. Last updated June 16, 2011. Available: http://www.its.dot.gov/eto/index.htm.

Vargo, S.L., and R.F. Lusch. 2004. "The Four Service Marketing Myths – Remnants of a Goods-Based Manufacturing Model." *Journal of Service Research*. 6 (4): 324-335.

Vest, C.M., 2013. "Educating Engineers for 2020 and Beyond" .*The Bridge*. Washington DC, National Academy of Engineering.

Wild, P.J., J. Jupp, W. Kerley, C. Eckert, and P.J. Clarkson. 2007. "Towards A Framework for Profiling of Products and Services." Presented at 5th International Conference on Manufacturing Research (ICMR), September 11-13, 2007, Leicester, UK.

Primary References

IFM. 2008. *Succeeding through Service Innovation: A service perspective for education, research, business and government*. University of Cambridge Institute for Manufacturing (IfM) and International Business Machines Corporation (IBM) report. Cambridge Service Science, Management and Engineering Symposium, July 14-15, 2007, Cambridge, UK.

Organization for Economic Co-operation and Development (OECD). 2000. *The Service Economy*. Science Technology Industry (STI) Business and Industry Policy Forum Series. Paris, France: OECD. Available: http:// www.oecd.org/dataoecd/10/33/2090561.pdf.

Vargo, S.L., and R.F. Lusch. 2004. "The Four Service Marketing Myths – Remnants of a Goods-Based Manufacturing Model." *Journal of Service Research*. 6 (4): 324-335.

Additional References

Andrikopoulos, V., A. Bucchiarone, E. Di Nitto, R. Kazhamiakin, S. Lane, V. Mazza, and I. Richardson. 2010. "Chapter 8: Service Engineering," in *Service Research Challenges and Solutions for the Future Internet S-Cube – Towards Engineering, Managing and Adapting Service-Based Systems*, edited by M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger. Berlin Heidelberg, Germany: Springer-Verlag. p. 271-337.

Cook, M. 2004. "Understanding The Potential Opportunities Provided by Service-Orientated Concepts to Improve Resource Productivity," in *Design and Manufacture for Sustainable Development 2004*, edited by T. Bhamra and B. Hon. Bury St. Edmonds, Suffolk, UK: Professional Engineering Publishing Limited. p. 123-134.

Domingue, J., D. Fensel, J. Davies, R. González-Cabero, and C. Pedrinaci. 2009. "The Service Web: a Web of Billions of Services," in *Towards the Future Internet - A European Research Perspective*, edited by G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zehariadis. Amsterdam, The Netherlands: IOS Press.

Maglio P., and J. Spohrer 2008. "Fundamentals of Service Science." *Journal of the Academy of Marketing Science*. 36 (1): 18-20. DOI: 10.1007/s11747-007-0058-9.

McAfee, A. 2009. *Enterprise 2.0: New Collaborative Tools for Your Organization's Toughest Challenges*. Boston, MA, USA: Harvard Business School Press.

Mont, O., and A. Tukker. 2006. "Product-Service Systems." *Journal of Cleaner Production*. 14 (17): 1451-1454.

Moran, M. 2006. *Servicizing Solar Panels.* Industry Course Report. Lund University International Master's Programme in Environmental Studies and Sustainability Science Department (LUMES), Lund University, Sweden.

Skyttner, L. 2006. *General Systems Theory: Perspectives, Problems, Practice*, 2nd ed. Singapore: World Scientific Publishing Company.

Spohrer, J., and S.K. Kwan. 2009. "Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline - Outline & References." *International Journal of Information Systems in the Service Sector*. 1 (3): 1-31.

Spohrer, J.C. 2011. "Service Science: Progress & Directions." Presented at the International Joint Conference on Service Science, May 25-27, 2011, Taipei, Taiwan.

TM Forum. 2008. *Service Delivery Framework (SDF) Overview, Release 2.0*. Morristown, NJ: TeleManagement Forum. Technical Report 139.

Tselentis, G., J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zahariadis (eds.). 2009. *Towards the Future Internet - A European Research Perspective*. Amsterdam, The Netherlands: IOS Press.

US DOT. 2011. "Emergency Transportation Operations." Research and Innovative Technology Administration. Accessed June 23, 2011. Last updated June 16, 2011. Available: http://www.its.dot.gov/eto/index.htm.

Wild, P.J., J. Jupp, W. Kerley, C. Eckert, and P.J. Clarkson. 2007. "Towards A Framework for Profiling of Products and Services." Presented at 5th International Conference on Manufacturing Research (ICMR), September 11-13, 2007, Leicester, UK.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Fundamentals of Services

Services are activities that cause a transformation of the state of an entity (a person, product, business, region, or nation) by mutually agreed terms between the service provider and the customer. Individual services are relatively simple, although they may require customization and significant back-stage support (e.g., database, knowledge management, analysis, forecasting, etc.) to assure quality and timely delivery. Product services are also relatively straightforward as product specifications, performance standards, quality control, installation guidelines, and maintenance procedures require good communication and understanding between providers and users. Business services can be rather complex; some may involve intensive negotiations, work process alignment, quality assurance, team collaboration, and service coproduction. Moreover, Chang (2010) states that: “Regional and National services are even more complex, as they may affect policy, custom regulations, export permits, local business practices, logistics, distribution, and other such issues" (see also Complexity).

Service Systems

The service and/or set of services developed and accessible to the customer (individual consumer or enterprise) are enabled by a service system. Service system stakeholders may interact to create a particular service value chain to be delivered with a specific objective (Spohrer and Maglio 2010). Service system entities dynamically configure four types of resources: people, technology/environment infrastructure, organizations(glossary)/institutions, and shared information/symbolic knowledge. Service systems can be either formal or informal in nature. In the case of formal service systems, the interactions are contracted through service level agreements (SLA). Informal service systems can promise to reconfigure resources without a written contractual agreement; in the case of the emergency transports operations example discussed in the Service Systems Background article, there is no formal contractual agreement (i.e., SLA) between the user requesting the service and the agency providing the service other than a “promise” for a quick and efficient response. SLAs are written contracts between and among service system entities, as well as the legal system for enforcing the contracts. The study of informal service systems contains the study of relationships (communications, interactions, and promises) between service systems and social systems, cultural norms and beliefs, as well as political systems that can maintain those relationships (Spohrer and Kwan 2008). The resources are either physical or non-physical and have rights or no rights. See Figure 1 below:

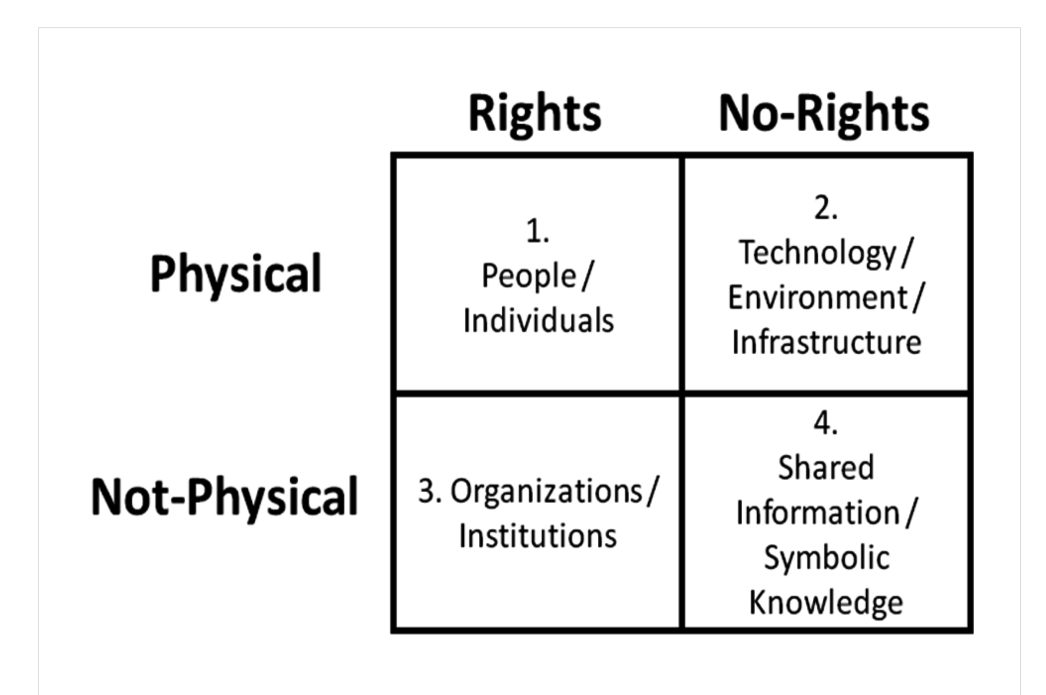


Figure 1. Service System Resources (Spohrer 2011). Reprinted with permission of Dr. James C. Spohrer. All other rights are reserved by the copyright owner.

Service Value Chain

SLAs and policies specify the conditions under which services system entities reconfigure access rights to resources by mutually agreed value propositions. Current management frameworks typically focus on single service system entity interfaces. They neither use SLAs for managing the implementation and delivery of services nor do they recognize/support the fact that many services may be composed of lower-level services, involve third-party providers, and rely on possibly complex relationships and processes among participating businesses, information communications, and technologies (CoreGRID 2007). While SLAs are mapped to the respective customer requirements, policies are provider-specific means to express constraints and rules for their internal operations. These rules may be independent of any particular customer (Theilmann 2009).

In service systems practice, we describe the service value chain in terms of links among the entities connected via the Network Centric operations of service systems. For instance, value could then be created and delivered in terms of e-services, such as business-to-business (B2B), business to consumer (B2C), business to government (B2G), government-to-business (G2B), government-to-government (G2G), government-to-consumer (G2C), etc. The emerging service in this case interacts or “co-produces” with their customer via the World Wide Web as compared to the physical environment in which the traditional, or brick and mortar, service enterprises interact with their customers.

The services sector requires information as input, involves the customer at the production/delivery stage, and employs mostly qualitative measures to assess its performance, i.e., technology-intensive services are “information-driven, customer centric, e-oriented, and productivity-focused" (Tien and Berg 2003; Hipel et al. 2007; Chesbrough 2011). Chang (2010) defines these features in this manner:

- **Information Driven:** The creation, management, and sharing of information is crucial to the design, production, and delivery of services.
- **Customer Centric:** Customers are generally the co-producer of the services, as in the case of self-service. Customers require a certain degree of self-adaptation or customization and customers must be satisfied with the rendered services.

- **E (electronics) Oriented:** Services are becoming increasingly e-oriented. Thus, e-access, e-commerce, and e-customer management are crucial to e-services.
- **Productivity Focused:** Both efficiency and effectiveness are important in the design, delivery, and support of services.
- **Value Adding:** Services need to provide some value for the target clients. For profit-seeking service companies, the value produced for customers assures the company's profitability. For non-profit service entities, the value produced for customers reinforces the quality of a service entity's policy.

A service system is defined by its value co-creation chain in which stakeholders work in open collaboration to deliver consistently high quality service according to business goals, service goals, and customer goals. A value proposition can be viewed as a request from one service system to another to run an algorithm (the value proposition) from the perspectives of multiple stakeholders according to culturally determined value principles. The four primary stakeholder’s perspectives in regards to value are the customer, provider, authority, and the competitors. Figure 2 below depicts value calculations from multiple stakeholder perspectives.

Table 1. Value Calculation from Different Stakeholders' Perspectives (Spohrer 2011). Reprinted with permission of Dr. James C. Spohrer. All other rights are reserved by the copyright owner.

Stakeholder Perspective (the players)	Measure Impacted	Pricing Decision	Basic Questions	Value Proposition Reasoning
1. Customer	Quality (Revenue)	Value Based	Should we? (offer it)	Model of customer: Do customers want it? Is there a market? How large? Growth rate?
2. Provider	Productivity (Profit, Mission, Continuous Improvement, Sustainability)	Cost Plus	Can we? (deliver it)	Model of self: Does it play to our strengths? Can we deliver it profitability to customers? Can we continue to improve?
3. Authority	Compliance (Taxes and Fines, Quality of Fire)	Regulated	May we? (offer and deliver it)	Model of authority: Is it legal? Does it compromise our integrity in any way? Does it create a moral hazard?
4. Competitor (Substitute)	Sustainable Innovation (Market Share)	Strategic	Will we? (invest to make it so)	Model of competitor: Does it put us ahead? Can we stay ahead? Does it differentiate us from the competition?

From an engineering design point of view, the service and business goals are an entry point through which to analyze the business architectures (including organization and processes) needed, which in turn demand alignment between the information technology (IT) components and technology architecture to achieve the goals. From a systems engineering perspective, the next step is to identify service system entities that could participate in the service delivery (people, organizations, technologies, processes, etc.).

Service System Entities

Spath and Fahnrich (2007) defined a service meta-model comprised of nine types of **entities**:

1. **Customers**: customer features, customer attitudes, and customer preferences;
2. **Goals**: business goals, service goals, customer goals, and enterprise culture goals;
3. **Inputs**: physical, human beings, information, knowledge, currency, and constraints;
4. **Outputs**: physical, human beings, information, knowledge, currency, and waste;
5. **Processes**: service provision, service operations, service support, customer relationships, planning and control, and call center management;
6. **Human Enablers**: service providers, support providers, management, and owner organization (enterprise);
7. **Physical Enablers**: owner organization (physical), buildings, equipment, furnishings, and location;
8. **Informatics Enablers**: information, knowledge, procedures and processes, decision support, and skill acquisition; and
9. **Environment**: political factors, economic factors, social factors, technological factors, environmental factors, legal factors (PESTEL), and physical factors.

Thus, a service or service offering is created by the relationships among service system entities (including information flows) through business processes into strategic capabilities that consistently provide superior value to the customer. If we were to represent the service as a network diagram (as in Figure 3 below), then the entities represent the nodes and the links represent the relationships between nodes.

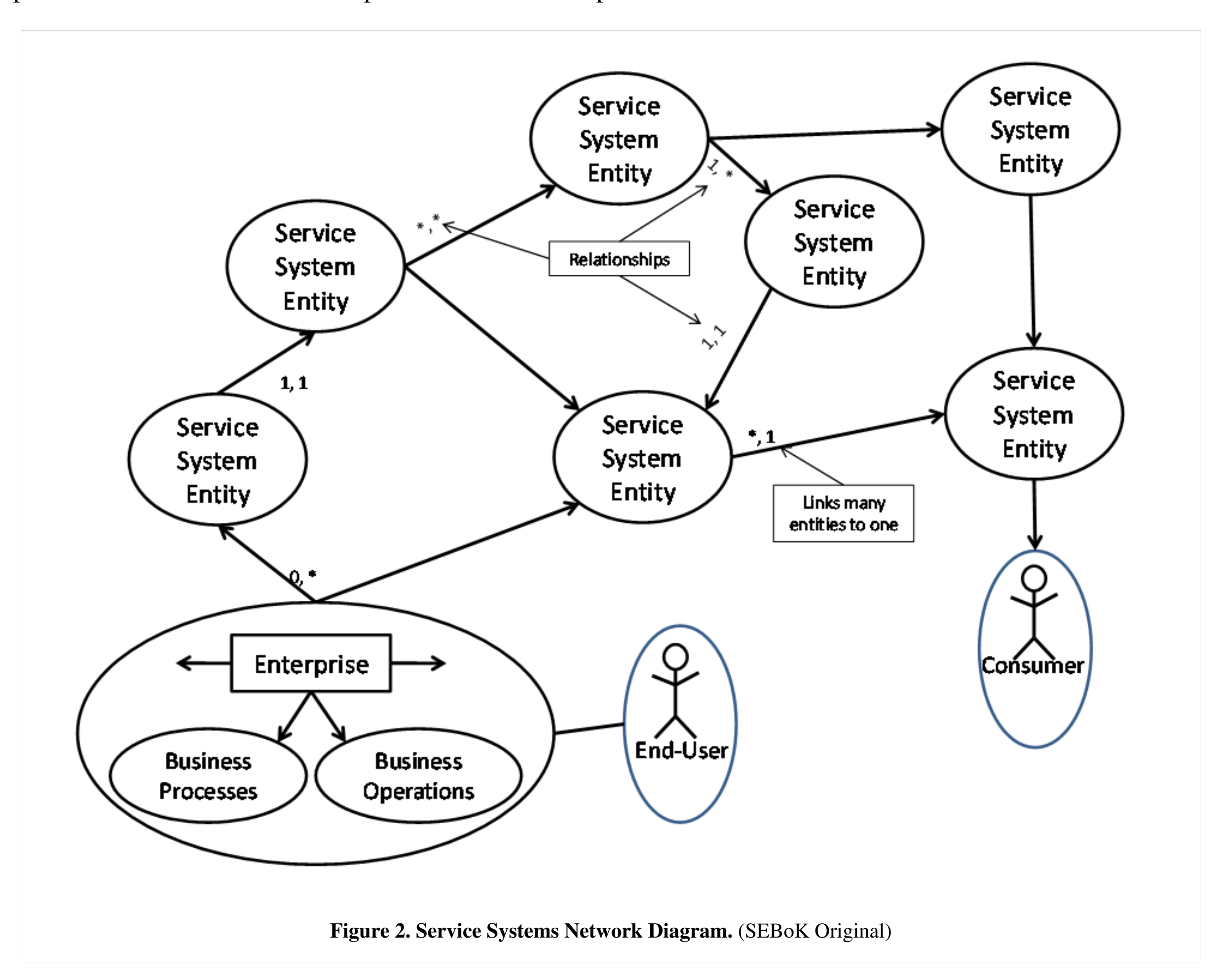


Figure 2. Service Systems Network Diagram. (SEBoK Original)

Service System Hierarchy

Systems are part of other systems which are often expressed by systems hierarchies (Skyttner 2010) to create a multilevel hierarchy, and thus the service system is composed of service system entities that interact through processes defined by governance and management rules to create different types of outcomes in the context of stakeholders with the purpose of providing improved customer interaction and value co-creation. Examples of service system entities are business enterprises, nations, or in the simplest form, a person (consumes and produces services).

Using the hierarchical approach, Spohrer conceptualizes an ecosystem at the highest level in which a service system is an entity of its own. This concept is extended to create the service system hierarchy as described in Figure 4 below (Spohrer 2011; Maglio and Spohrer 2008; Maglio et al. 2010).

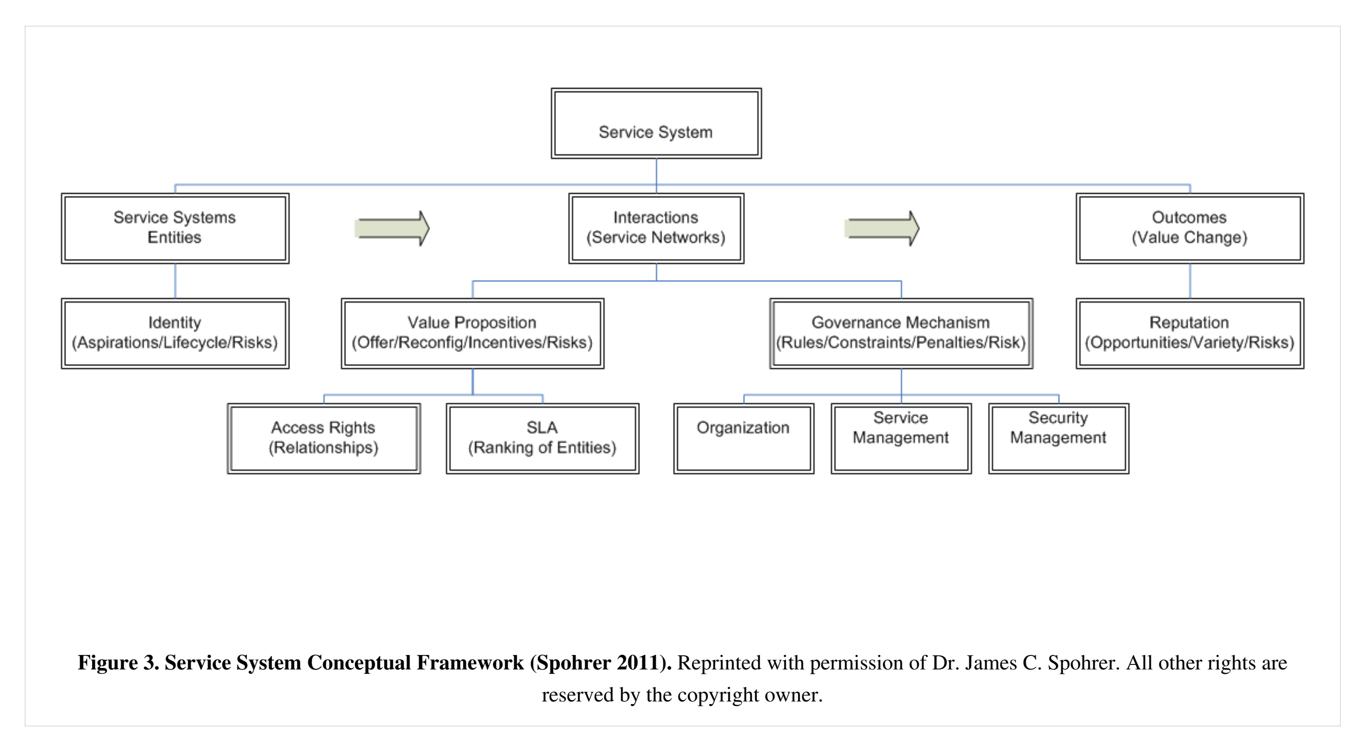


Figure 3. Service System Conceptual Framework (Spohrer 2011). Reprinted with permission of Dr. James C. Spohrer. All other rights are reserved by the copyright owner.

Service System Attributes

The fundamental attributes of a service system include togetherness, structure, behavior, and emergence. As mentioned earlier, today's global economy is very competitive and a service system may be very competitive in a given environment at a given time (the business space). The service system's trajectory should be well controlled as time goes by (Qiu 2009) since services are "real time in nature and are consumed at the time they are co-produced" (Tien and Berg 2003), that is, during service transactions.

The service system should evolve and adapt to the conditions within the business space in a manner which ensures that the customized service behaves as expected. This adaptive behavior of service systems implies that their design must be truly trans-disciplinary:

> *They must include techniques from social science (i.e., sociology, psychology, and philosophy) and management (i.e., organization, economics, and entrepreneurship). As a consequence, Systems, Man, and Cybernetics (SMC) must expand their systems (i.e., holistic oriented), man (i.e., decision-oriented), and cybernetics methods to include and be integrated with those techniques that are beyond science and engineering.* (Hipel et al. 2007)

References

Works Cited

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence*. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Chesbrough, H. 2011. *Open Services Innovation: Rethinking Your Business to Grow and Compete in a New Era*. San Francisco, CA, USA: Jossey-Bass.

CoreGRID. 2007. *Using SLA for Resource Management and Scheduling - A Survey. Technical Report 0096*. Jülich & Dortmund, Germany: European Research Network on Foundations, Software Infrastructures and Applications for Large Scale Distributed, GRID and Peer-to-Peer Technologies, Institute on Resource Management and Scheduling. Accessed June 4, 2011. Available: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0096.pdf.

Hipel, K.W., M.M. Jamshidi, J.M. Tien, and C.C. White. 2007. "The Future of Systems, Man, and Cybernetics: Application Domains and Research Methods." *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*. 37 (5): 726-743.

Maglio, P., C. Kieliszewski, and J. Spohrer. 2010. *Handbook of Service Science.* New York, NY, USA: Springer Science and Business Media.

Maglio, P., and J. Spohrer. 2008. "Fundamentals of Service Science." *Journal of the Academy of Marketing Science*. 36 (1): 18-20.

Qiu, R. 2009. "Computational Thinking of Service Systems: Dynamics and Adaptiveness Modeling." *Service Science*. 1 (1): 42-55.

Skyttner, L. 2006. *General Systems Theory: Perspectives, Problems, Practice,* 2nd ed. Singapore: World Scientific Publishing Company.

Spath, D., and K.P. Fähnrich (eds.). 2007. *Advances in Services Innovations*. Berlin & Heidelberg, Germany: Springer-Verlag.

Spohrer, J., and S.K. Kwan. 2009. "Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline - Outline & References." *International Journal of Information Systems in the Service Sector*, 1 (3): 1-31.

Spohrer, J., and P.P Maglio. 2010. "Chapter 1: Service Science: Toward a Smarter Planet," in *Introduction to Service Engineering*, edited by G. Salvendy and W. Karwowski. Hoboken, NJ: John Wiley & Sons.

Spohrer, J.C. 2011. "Service Science: Progress & Directions." Presented at International Joint Conference on Service Science, May 25-27, 2011, Taipei, Taiwan.

Theilmann, W., and L. Baresi. 2009. "Multi-level SLAs for Harmonized Management in the Future Internet," in *Towards the Future Internet - A European Research Perspective*, edited by G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zehariadis. Amsterdam, The Netherlands: IOS Press.

Tien, J.M., and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering*. 12 (1): 13-38.

Vargo, S.L., and M.A. Akaka. 2009. "Service-Dominant Logic as a Foundation for Service Science: Clarifications." *Service Science*. 1 (1): 32-41.

Primary References

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence*. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Hipel, K.W., M.M. Jamshidi, J.M. Tien, and C.C. White. 2007. "The Future of Systems, Man, and Cybernetics: Application Domains and Research Methods." *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*. 37 (5): 726-743.

Spath, D., and K.P. Fähnrich (eds.). 2007. *Advances in Services Innovations*. Berlin & Heidelberg, Germany: Springer-Verlag.

Spohrer, J.C. 2011. "Service Science: Progress & Directions." Presented at International Joint Conference on Service Science, May 25-27, 2011, Taipei, Taiwan.

Additional References

Chesbrough, H. 2011. *Open Services Innovation: Rethinking Your Business to Grow and Compete in a New Era*. San Francisco, CA, USA: Jossey-Bass.

CoreGRID. 2007. *Using SLA for Resource Management and Scheduling - A Survey. Technical Report 0096*. Jülich & Dortmund, Germany: European Research Network on Foundations, Software Infrastructures and Applications for Large Scale Distributed, GRID and Peer-to-Peer Technologies, Institute on Resource Management and Scheduling. Accessed June 4, 2011. Available: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0096.pdf.

Maglio, P., C. Kieliszewski, and J. Spohrer. 2010. *Handbook of Service Science.* New York, NY, USA: Springer Science and Business Media.

Maglio, P., and J. Spohrer. 2008. "Fundamentals of Service Science." *Journal of the Academy of Marketing Science*. 36 (1): 18-20.

Qiu, R. 2009. "Computational Thinking of Service Systems: Dynamics and Adaptiveness Modeling." *Service Science*. 1 (1): 42-55.

Skyttner, L. 2006. *General Systems Theory: Perspectives, Problems, Practice,* 2nd ed. Singapore: World Scientific Publishing Company.

Spohrer, J., and S.K. Kwan. 2009. "Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline - Outline & References." *International Journal of Information Systems in the Service Sector*, 1 (3): 1-31.

Spohrer, J., and P.P Maglio. 2010. "Chapter 1: Service Science: Toward a Smarter Planet," in *Introduction to Service Engineering*, edited by G. Salvendy and W. Karwowski. Hoboken, NJ: John Wiley & Sons.

Theilmann, W., and L. Baresi. 2009. "Multi-level SLAs for Harmonized Management in the Future Internet," in *Towards the Future Internet - A European Research Perspective*, edited by G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zehariadis. Amsterdam, The Netherlands: IOS Press.

Tien, J.M., and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering*. 12 (1): 13-38.

Vargo, S.L., and M.A. Akaka. 2009. "Service-Dominant Logic as a Foundation for Service Science: Clarifications." *Service Science*. 1 (1): 32-41.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Properties of Services

A service is realized by the service system through the relationships of service system entities that interact (or relate) in a particular way to deliver the specific service via a service level agreement (SLA). Current management frameworks typically only focus on the interfaces of single service system entities. Meanwhile, SLAs are mapped to the respective customer requirements. These policies are provider-specific means to express constraints and rules for their internal operations. These rules may be independent of any particular customer (Theilmann 2009).

Services not only involve the interaction between the service provider and the consumer to produce value, but have other attributes, like an intangible quality of service (e.g., an ambulance service's availability and response time to an emergency request). The demand for a service may have varying loads dependent on the time of day, day of week, season, or other unexpected needs (e.g., natural disasters, product promotion campaigns, etc.). In the US for instance, travel services have peak demands during Christmas week; Mother's day is usually the highest volume handling day for a telecommunications provider and tax services peak during extended periods (January through mid-April). Services cannot be inventoried; they are rendered at the time they are requested.

Additionally, for a business enterprise, delivering the service at the minimum cost while maximizing its profits may be the service objective. In contrast, for a non-profit organization the objective may be to maximize customer satisfaction while optimizing the resources required to render the service (e.g., during a natural disaster). Thus, the design and operations of service systems "is all about finding the appropriate balance between the resources devoted to the systems and the demands placed on the system so that the quality of service to the customer is as good as possible" (Daskin 2010).

Service Level Agreement

A SLA is a set of technical (functional) and non-technical (non-functional) parameters agreed among customers and service providers. SLAs can and do contain administrative level (non-functional) business related parameters, such as SLA duration, service availability for the SLA duration, consequences for variations, failure reporting, priorities, and provisions for modifications to the SLA. However, for service level management, the service level (technical) parameters need to be defined, monitored, and assessed; these parameters may include such things as throughput; quality; availability; security; performance; reliability, for example, mean time between failure (MTBF), maximum downtime, and time-to-repair; and resource allocation.

An SLA represents the negotiated service level requirements (SLR) of the customer and should establish valid and reliable service performance measures since it is usually the basis for effective service level management (SLM). The goal of SLM is to ensure that service providers meet and maintain the prescribed quality of service (QoS). However, care should be taken since in some domains the term QoS refers only to resource reservation control mechanisms rather than the achieved service quality (e.g., internet protocol (IP) networks). Some terms used to mean the "achieved service quality" include quality of experience (QoE), user-perceived performance, and degree of satisfaction of the user; these other terms are more generally used across service domains.

Non-functional properties fall into two basic categories: business properties, such as price and method of payment, and environmental properties, such as time and location. Business and environmental properties are classified as "context properties" by Youakim Badr (Badr et al. 2008). QoS properties are characteristics such as availability, resilience, security, reliability, scalability, agreement duration, response times, repair times, usability, etc. Therefore services evaluation measures are customer oriented and include not only traditional performance metrics

(productivity, quality, etc.), but also require a comprehensive analysis of the service system from an end-to-end perspective. Service evaluation typically includes customer demand-supply to ensure economic viability across the lifecycle of the service system. Furthermore, the service delivery is evaluated using the key technical performance metrics listed above, adding also Service Process Measures (provisioning time, time-to-restore/repair, etc.) and Technical Performance Measures (end-to-end response times, latency, throughput, etc.). Finally, the service system's SLAs are then the composition of these categories evaluated on a systemic level to ensure consistency, equity, and sustainability of the service to assure that the desired/contracted SLA for customer satisfaction, value co-creation, and high system robustness are realized. (Spohrer 2011; Tien and Berg 2003; Theilmann and Baresi, 2009)

Service Key Performance Indicators

Service key performance indicators (KPI) are defined and agreed to in the SLA; the service KPIs are decomposed into service process measures (SPM) and technical performance measures (TPM) during the analysis stage of the service systems engineering (SSE) process. In the design process, the KPIs and TPM are allocated to service system entities and their components, as well as to the business processes and their components so as to ensure compliance with SLAs. The allocated measures generate derived requirements (SLR) for the system entities and their relationships, as well as for the service entities' components and the data and information flows required in the service systems to monitor, measure, and assess end-to-end SLA. These allocations ensure that the appropriate performance indicators apply to each of the links in the service value chain.

TPMs are typically categorized by the number of defective parts in a manufacturing service, data transmission latency and data throughput in an end-to-end application service, IP QoS expressed by latency, jitter delay, and throughput; SPMs are typically categorized by service provisioning time, end-to-end response times to a service request (a combination of data and objective feedback), and quality of experience (QoE verified by objective feedback). Together, the KPI (TPM combined with SPM) and perception measures make up the service level management function. A quality assurance system's (QAS) continuous service improvement (CSI), processes, and process quality management and improvement (PQMI) should be planned, designed, deployed, and managed for the capability to continuously improve the service system and to monitor compliance with SLAs (e.g., PQMI, capability maturity model integration (CMMI) (SEI 2007), International Organization for Standardization (ISO) Standards 9001 (ISO/IEC 2008), Telecom Quality Management System Standards (TL 9000) (QuEST Forum 2012), Information Technology Infrastructure Library (ITIL) v. 3 (OGC 2009), etc.).

As discussed earlier, QoS needs to correlate customer perceived quality (subjective measures) with objective SPM and TPM measures. There are several techniques available to help monitor, measure, and assess TPM's, but most are a variation on the theme of culling information from TPM's using, for example, perceptual speech quality measure (PSQM) and perceptual evaluation of video quality (PEVQ) and enhancing or verifying this information with customer or end-user perception of service by extending mean opinion score (MOS) techniques/customer opinion models (Ray 1984). Telecommunication systems engineering (TCSE) played an important role in finding methodologies for correlation between perception and objective measures for the services of the twentieth century; SSE should continue to encourage multidisciplinary participation to equally find methodologies, processes, and tools to correlate perceived service quality with TPM and with SPM for the services of the twenty-first century (Freeman 2004).

Subjective (qualitative) service quality is the customer's perceived conformity of the service with the expected objective. Word-of-mouth, personal needs, and past experiences create customer expectations regarding the service. The customers' perception of the service must be captured via surveys and interviews. The customers' perception of the service is then compared with their expectations for the service; this process captures the perceived service quality. Care should be taken to understand that subjective measures appear to measure customer attitudes, and attitudes may be the result of several encounters with the service, as well as numerous encounters with similar services.

In summary, the SLA documents the SLRs and establishes reliable and valid service performance measures, technical parameters, and the agreed performance levels for the technical parameters. The technical parameters are then monitored and continuously compared against both objective and subjective data culled from multiple internal and external sources (service level management). The goal is not to report the level of service in a given period, but to develop and implement a dynamic system capable of predicting and driving service level improvement over time (i.e., continual service improvement (CSI)).

Evolution of Services

The second, third, and fourth decades of the twenty-first century will almost certainly see similar, and probably accelerated, technology development as seen in the prior three decades. Mass collaboration will become an established mode of operation. The beginnings of mass collaboration have manifested in developments such as value co-creation where loosely entangled actors or entities come together to create value in unprecedented ways, but ways that meet mutual and broader market requirements. Further developments in the technology, use, and acceptance of social media will continue to fuel the acceleration of these developments.

The next decades will see the grounding of concepts, such as crowdsourcing, coined by Jeff Howe in a June 2006 *Wired* magazine article; open innovation, promoted by Henry Chesbrough, a professor and executive director at the Center for Open Innovation at Berkeley; and mass collaboration and open source innovation supported by Enterprise 2.0 tools, as conceived by Wikinomics consultant Don Tapscott.

Roberto Saracco, a telecommunications expert specializing in analyzing economical impacts of technology evolution, argues that: “Communications will be the invisible fabric connecting us and the world whenever and wherever we happen to be in a completely seamless way, connecting us so transparently, cheaply, and effortlessly that very seldom will we think about it.” The ubiquity and invisibility of these communications will greatly facilitate the creation and destruction of ad hoc collectives (groups of entities that share or are motivated by at least one common issue or interest, or work together on a specific project(s) to achieve a common objective). This enterprise may engender the concept of the hive mind (the collective intelligence of many), which will be an intelligent version of real-life super organisms, such as ant or bee nests (Hölldobler and Wilson 2009).

These models will most certainly give rise to issues of property rights and liabilities; access rights for both the provider and the customer can be owned outright, contracted/leased, shared, or have privileged access (Spohrer 2011). For now, we are on the cusp of a management revolution that is likely to be as profound and unsettling as the one that gave birth to the modern industrial age. Driven by the emergence of powerful new collaborative technologies, this transformation will radically reshape the nature of work, the boundaries of the enterprise, and the responsibilities of business leaders (McAfee 2009).

The service-providing industry in the US is divided into thirteen sectors (Chang 2010):

1. professional and business services,
2. healthcare and social assistance,
3. state and local government,
4. leisure and hospitality,
5. other services,
6. educational services,
7. retail trade,
8. financial activities,
9. transportation and warehousing,
10. wholesale trade,
11. information,
12. federal government, and
13. utilities.

Spohrer (2011) goes beyond the service sectors to propose three types of service systems:

1. ***Systems that focus on flow of things***: transportation and supply chains, water and waste recycling, food and products, energy and electric Grid, information/ICT & cloud;
2. ***Systems that focus on Human Activities and Development***: buildings and construction, retail and hospitality / media and entertainment industries, banking and finance / business consulting industries, healthcare and family life systems, education and work life / jobs and entrepreneurship; and
3. ***Systems that focus on Governing***: cities, states, and nations.

Categorizing types and sectors of services is an important beginning because it can lead to a better understanding of the emerging rules and relationships in service value chains. This approach can further enhance the value co-creation capabilities of innovative service concepts that contribute to our quality of life. The classification also helps in identifying different objectives and constraints for the design and operations of the service system. Some examples include strategic policies under limited budget: education, strategic with readiness for quick response; national defense; business enterprise, maximizing profit while minimizing cost; etc.

In addition, this classification is being used to determine the overlap and synergies required among different science disciplines to enable trans-disciplinary collaboration and educational programs.

References

Works Cited

Badr, Y., A. Abraham, F. Biennier, and C. Grosan. 2008. "Enhancing Web Service Selection by User Preferences of Non-Functional Features." Presented at 4th International Conference on Next Generation Web Services Practices, October 20-22, 2008, Seoul, South Korea.

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence*. New York, NY, USA: John Wiley & Sons, Inc.

Daskin, M.S. 2010. *Service Science*. New York, NY, USA: John Wiley & Sons.

Freeman, R.L. 2004. *Telecommunication Systems Engineering*, 4th ed. New York, NY, USA: John Wiley & Sons.

Hölldobler, B., and E.O. Wilson. 2009. *The Super-organism: The Beauty, Elegance, and Strangeness of Insect Societies*. New York, NY, USA: W.W. Norton & Company.

ISO. 2008, ISO 9001:2008, *Quality management systems -- Requirements*. Geneva, Switzerland: International Organisation for Standardisation.

McAfee, A. 2009. *Enterprise 2.0: New Collaborative Tools for Your Organization's Toughest Challenges*. Boston, MA, USA: Harvard Business School Press.

OGC (Office of Government Commerce). 2009. *ITIL Lifecycle Publication Suite Books*. London, UK: The Stationery Office.

QuEST Forum. 2012. *Quality Management System (QMS) Measurements Handbook*, Release 5.0. Plano, TX, USA: Quest Forum.

Ray, R.F. (ed). 1984. *Engineering and Operations in Bell System*, 2nd ed. Florham Park, NJ, USA: AT&T Bell Labs.

SEI. 2007. *Capability Maturity Model Integrated (CMMI) for Development*, version 1.2. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Spohrer, J.C. 2011. "Service Science: Progress & Directions." Presented at the International Joint Conference on Service Science, 25-27 May 2011, Taipei, Taiwan.

Theilmann, W., and L. Baresi. 2009. "Multi-level SLAs for Harmonized Management in the Future Internet," in *Towards the Future Internet - A European Research Perspective*, edited by G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zehariadis. Amsterdam, The Netherlands: IOS Press.

Tien, J.M., and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering.* 12 (1): 13-38.

Primary References

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* New York, NY, USA: John Wiley & Sons, Inc.

Theilmann, W., and L. Baresi. 2009. "Multi-level SLAs for Harmonized Management in the Future Internet," in *Towards the Future Internet - A European Research Perspective*, edited by G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zehariadis. Amsterdam, The Netherlands: IOS Press.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Scope of Service Systems Engineering

Service systems engineering (SSE) involves all aspects of the enterprise. This topic discusses different aspects of the scope of SSE, from organizational strategy, to interoperability, to the life cycle of services, and then to their design.

SSE and the Enterprise

Enterprises plan, develop, and manage the enhancements of their infrastructure, products, and services, including marketing strategies for product and service offerings. These plans propose new products or service offerings based on new, unexplored, or unforeseen customer needs with clearly differentiated value propositions. Service strategies are the internal business processes required to design, operate, and deliver services. The mission of service strategies is to develop the capacity to achieve and maintain a strategic advantage (OGC 2009).

Taking the systems engineering (SE) approach to service systems, or (SSE), is imperative for the service-oriented, customer-centric holistic view to select and combine service system entities. The SSE approach can then define and discover relationships among service system entities to plan, design, adapt, or self-adapt to co-create value. The SSE approach should identify linkages, relationships, constraints, challenges/problems, new technologies, interoperability standards, interface agreements, or process development requirements among service entities required for the planned service or for potential future services (Lefever 2005).

SSE mandates participation not only from engineering, business operations, and customers, but also from various different domains, such as management science, behavioral science, social science, systems science, network science, computer science, decision informatics, etc.

Hipel et al. (2007) have presented a table for service science in terms of the domains and methods, including not only service systems, but also infrastructure and transportation systems, environmental and energy systems, and defense and space systems. The collaboration domains in Figure 1 below are a first approximation to the collaboration required from different disciplines for the SSE paradigm.

Table 1. Service Systems Engineering Domain Collaboration. (Hipel et al. 2007) Reprinted with permission of © Copyright IEEE - All rights reserved.

SEE	Collaboration Domains
SSE Management	• Management Science • Business Process Management • Cognitive Science • Decision Science
Service Realization Process (SRP)	• All engineering fields • Business Operations • Infrastructure Operations • Social Science • Computer Science • Management Science • Behavioral Science • Network Science • Computational Science • Systems Science • Decision Science
Methodologies, Processes, and Tools (MPT)	• Natural Science • Business Science (BPMN) • Mathematical • All engineering fields

Major challenges faced by SSE include the dynamic nature of service systems evolving and adapting to constantly changing operations and/or business environments, and the need to overcome silos of knowledge. Interoperability of service system entities through interface agreements must be at the forefront of the SSE design process for the harmonization of operations, administration, maintenance, and provisioning procedures of the individual service system entities (Pineda 2010).

In addition, service systems require open collaboration among all stakeholders, but recent research on mental models of multidisciplinary teams shows integration and collaboration into cohesive teams has proven to be a major challenge (Carpenter et al. 2010) (See also Team Dynamics). Thus, the emphasis on multidisciplinary (e.g., scientific, engineering, management, and social) education and training programs required to foster systems thinking helps bridge the gaps created by these silos of knowledge.

In the SSE approach, the social, governance, business, service, operations, and management activities are linked together through the service life cycle; service systems are by themselves a type of system of systems (SoS) where traditional systems engineering (TSE) practices need to be extended to include service systems entities' relationships (e.g., interface agreements among people, organizations, processes, and technologies) through information flows, technical interoperability, governance, and access rights within a system of systems.

Interoperability of Services

Interoperability among the different service system entities becomes highly relevant in SSE since the constituent entities are designed according to stakeholder needs; the entity is usually managed and operated to satisfy its own objectives independently of other system entities. The objectives of individual service system entities may not necessarily converge with the overall objectives of the service system. Thus, the need to include the following in the definition of a service system: analysis and design of the service system, governance frameworks to align political objectives, service strategies, business objectives, information and communications technologies (ICT) objectives, technology objectives and end-to-end operations, administration and maintenance procedures, and allocation of these procedures to individual entities (Luzeaux and Ruault 2010).

The previous discussion relates to a new service system development. There may be instances where a service is planned for delivery in phases of deployment (transition/deployment phase), or as presented earlier, if there is already a service system defined and deployed, then it's possible that the new request is for a service based application (SBA), in which case, the process is more focused on the adaptations needed to deploy the new application. For SBA, instances of advances in computer engineering, computer science, and software development already permit the adaptation and creation of SBA in a run-time environment for the discovery, development, and publishing of applications (Maglio et al. 2010).

The service design process (SDP) for new services is triggered by the market concept of the intended service and considers the stakeholder(s), service value chain(s), target market(s), target customer(s), proposed SLA, demand forecast, pricing strategy, and customer access privileges, which together comprise the service strategy. The SDP process then adapts the TSE as a life cycle approach (concept/definition, design/development, deployment/transition, operations, life cycle management/utilization/CSI, and retirement) as discussed in Life Cycle Models. A more detailed list of the SSE process activities is described in Value of Service Systems Engineering and Service Systems Engineering Stages.

Service Lifecycle Stages

The SDP stages and notation are depicted in Table 2 below; due to the complexity of service systems (see also Complexity) the documents generated are becoming more model-based electronic documents than written binders depending on the methodologies and tools used.

Table 2. Service Realization Process: Life Cycle Stages. (SEBoK Original)

<html>

<table>
<tr><th colspan="2">Life Cycle Changes</th><th>Purpose</th><th>Decision Gates</th></tr>
<tr><td rowspan="3">Service Strategy/Concept</td><td>New Service identification</td><td rowspan="3">Elicit enterprise needs
Explore service concepts
Identify service system entities
Propose viable HL black box solutions
Output: Service Description</td><td rowspan="8">Decision Options
- Go, No-GO
- Continue this stage
- Go to preceding stage
- Hold project activity
- Terminate project
- Test
- Deploy</td></tr>
<tr><td>Feasibility Phase</td></tr>
<tr><td>HL Analysis</td></tr>
<tr><td rowspan="3">Service Design/Development</td><td>Service Requirement Analysis and Engineering</td><td rowspan="3">Refine service system requirements
Output: Service Requirement Document
Create solution description
Identify Interfaces among entities
Output: Preliminary Design
Develop service system detailed architecture and specs
Output: Service Specification Document
Verify and Validate system requirements
Output: service JV & V Plans</td></tr>
<tr><td>Service Development</td></tr>
<tr><td>Service Integration, Verification, and Validation</td></tr>
<tr><td colspan="2">Service Transition/Deployment</td><td>Service Insertion Plans
Deploy service system
Manage deployment activities
Inspect and test (verify)
Output: Service Operation Plans, Operations Technical Plans, Operational Readiness Plans</td></tr>
<tr><td colspan="2">Service Operations and / Continuous Service Improvement</td><td>Operate a reliable service system to satisfy customer needs
Monitor, Measure, & Assess
Provide sustained system capability
Troubleshoot potential issues
Store, archive, or dispose of the service system</td></tr>
</table>

</html>

All the life cycle stages are included for completeness, but very often during the concept analysis phase it may be determined that not all of the stages are needed. In these cases, a recommendation should be made regarding which stages are specifically required for the realization of the service in question.

Service Design Management

Another important role of SSE is the management of the service design process. SSE utilizes TSE practices to manage the resource and asset allocation to perform the activities required to realize the service through the value chain for both the customer and the service provider. The main focus of the service design process management is to provide for the planning, organizational structure, collaboration environment, and program controls to ensure that stakeholder's needs are met from an end-to-end customer perspective.

The service design process management process aligns business objectives and business operational plans with end-to-end service objectives, including customer management plans, service management and operations plans, and operations technical plans. The main SSE management activities are

- planning;
- assessment and control;
- decision management;
- risk management;
- configuration management; and
- information management.

SSE plays a critical role in describing the needs of the intended service in terms of the service's day-to-day operations, including customer care center requirements, interface among service system entities, such as: manufacturing plant, smart grid, hospital, network infrastructure provider(s), content provider(s) and service provider(s), service based application provider(s), applications providers, and the customer management process for the service.

Current research in computer engineering and software systems engineering is looking at the development of run-time platforms to allow real time or near real time customer service discovery and publishing (Spark 2009). The service-centric systems engineering (SeCSE) consortium has a well-defined service design process that is being applied to SBA. In this approach, there are design time and run-time sub-processes for the composition, provisioning, orchestration, and testing for service publishing (Lefever 2005). There is particular interest from the research community to include human-computer interactions (HCI) and behavioral science to address current social networking services (Facebook, Twitter, Linkedin, Google+, etc.) used to share unverified information via audio, messaging, video, chats, etc.

This research is gaining relevance because of the thin line between the customer (consumer, enterprise) and content providers in regards to security, privacy, information authentication, and possible misuse of the user-generated content. Even as the research progresses, these networking services are examples of business models organizing communities of interest for innovation. Hsu says, “If we understand this networking, then we may be able to see through the business strategies and systems design laws that optimize connected value co-creation” (2009).

References

Works Cited

Carpenter, S., H. Delugach, L. Etzkorn, J. Fortune, D. Utley, and S. Virani. 2010. "The Effect of Shared Mental Models on Team Performance." Presented at Industrial Engineering Research Conference, Institute of Industrial Engineers, 2010, Cancun, Mexico.

Hipel, K.W., M.M. Jamshidi, J.M. Tien, and C.C. White. 2007. "The Future of Systems, Man, and Cybernetics: Application Domains and research Methods." *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*. 37 (5): 726-743.

Hsu, C. 2009. "Service Science and Network Science." *Service Science*, 1 (2): i-ii.

Lefever, B. 2005. *SeSCE Methodology*. Rome, Italy: SeCSE Service Centric Systems Engineering. SeCSE511680. Available: http://www.secse-project.eu/wp-content/uploads/2007/08/a5_d4-secse-methodology-v1_3.pdf.

Luzeaux, D., and J.R. Ruault (eds.). 2010. *Systems of Systems*. New York, NY, USA: John Wiley & Sons.

Maglio, P., M. Weske, J. Yang, and M. Fantinato (eds.). 2010. *Proceedings of the 8th International Conference on Service Oriented Computing: ICSOC 2010.* Berlin & Heidelberg, Germany: Springer-Verlag.

OGC (Office of Government Commerce). 2009. *ITIL Lifecycle Publication Suite Books*. London, UK: The Stationery Office.

Pineda, R. 2010. "Understanding Complex Systems of Systems Engineering." Presented at Fourth General Assembly Cartagena Network of Engineering, 2010, Metz, France.

Spark, D. 2009. "Real-Time Search and Discovery of the Social Web." Spark Media Solutions Report. Accessed September 2, 2011. Available: http:/ / www. sparkminute. com/ 2009/ 12/ 07/ free-report-real-time-search-and-discovery-of-the-social-web/.

Primary References

Lefever, B. 2005. *SeSCE Methodology*. Rome, Italy: SeCSE Service Centric Systems Engineering. SeCSE511680. Available: http://www.secse-project.eu/wp-content/uploads/2007/08/a5_d4-secse-methodology-v1_3.pdf.

Luzeaux, D., and J.R. Ruault (eds.). 2010. Systems of Systems. New York, NY, USA: John Wiley & Sons.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Value of Service Systems Engineering

Service systems engineering (SSE) is a multidisciplinary approach to manage and design value (glossary) co-creation of a service system. It extends the holistic view of a system (glossary) to a customer-centric, end-to-end view of service system design. Service systems engineers must play the role of an integrator by considering the interface requirements for the interoperability of service system entities, not only for technical integration, but also for the processes and organization required for optimal customer experience during service operations.

Service systems engineering uses disciplined approaches to minimize risk by coordinating/orchestrating social aspects, governance (glossary) (including security), environmental, human behavior, business, customer care, service management, operations, and technology development processes. Therefore, systems engineers must have a good understanding of cross disciplinary issues to manage, communicate, plan, and organize service systems development and delivery of service. Service systems engineering also brings a customer focus to promote service excellence and to facilitate service innovation through the use of emerging technologies to propose creation of new service systems and value co-creation.

The service design process includes the definition of methods, processes, and procedures necessary to monitor and track service requirements verification (glossary) and validation, in particular as they relate to the operations, administration, maintenance, and provisioning procedures of the whole service system and its entities. These procedures ensure that failures by any entity are detected and do not propagate and disturb the operations of the service (Luzeaux and Ruault 2010).

Research on service systems needs to fuse business process management, service innovation, and social networks for the modeling of service system value chain (Carroll et al. 2010). The systems engineering approach helps to better understand and manage conflict, thereby helping both private and public organizations optimize their strategic decision making. The use of a systemic approach reduces rework, overall time to market, and total cost of development.

Service SE Knowledge & Skills

The world's economies continue to move toward the creation and delivery of more innovative services. To best prepare tomorrow's leaders, new disciplines are needed that include and ingrain different skills and create the knowledge to support such global services. "In this evolving world, a new kind of engineer is needed, one who can think broadly across disciplines and consider the human dimensions that are at the heart of every design challenge" (Grasso and Martinelli 2007).

Service systems engineers fit the T-shaped Model (glossary) of professionals (Maglio and Spohrer 2008) who must have a deeply developed specialty area, as well as a broad set of skills and capabilities (See the Enabling Individuals article). Chang (2010) lists the following twelve service system management and engineering (SSME) skills:

1. Management of Service Systems. These skills include scheduling, budgeting and management of information systems/technologies, and leadership;
2. Operations of Service Systems. Engineers should be proficient in process evaluation and improvement, Quality (glossary) improvement, customer relationships, and Uncertainty (glossary) management;
3. Service Processes. These skills include performance measurements, flow charting, work task breakdown;
4. Business Management. Business skills include project costing, business planning, and change management;
5. Analytical Skills. These skills include problem solving, economic decision analysis, risk analysis, cost estimating, probability and statistics;
6. Interpersonal Skills. Increasingly, service systems engineers are expected to excel in professional responsibility, verbal skills, technical writing, facilitating, and team building;

7. Knowledge Management. Service systems engineers should be familiar with definition, strategies, success factors, hurdles, and best practices in industry;
8. Creativity and Innovation in Services. These skills include creative thinking methods, success factors, value chain, best practices, and future of innovation;
9. Financial and Cost Analysis and Management. Additional business skills include activity-based costing, cost estimation under uncertainty, T-account, financial statements, ratio analysis, balanced scoreboards, and capital formation;
10. Marketing Management. Market forecast, market segmentation, marketing mix- service, price, communications and distribution- are important marketing tools;
11. Ethics and Integrity. Service Systems Engineers must be held to high ethical standards. These include practicing ethics in workplace and clear guidelines for making tough ethical decisions, corporate ethics programs, affirmation action, and workforce diversity, as well as global issues related to ethics. (See Ethical Behavior); and
12. Global Orientation. Increasingly, engineers must be aware of emerging business trends and challenges with regards to globalization drivers, global opportunities, and global leadership qualities.

Service Architecture, Modeling & Views

Successful deployment of service value chains is highly dependent on the alignment of the service with the overall enterprise service strategy, customer expectations, and customer's service experience. The importance of service-oriented customer-centric design has been recognized for several years by traditional service providers (telecommunications, information technology (IT), business reengineering, web services, etc.) through the creation of process-driven architecture frameworks.

Architecture frameworks are important for creating a holistic system view. They promote a common understanding of the major building blocks and their interrelation in systems of systems or complex systems of systems (see also Complexity). An architecture (glossary) is a model of the the system created to describe the entities, the interactions and interoperability among entities, as well as the expected behavior, utilization, and properties of the end-to-end system. The architectures become the main tool to guide stakeholders, developers, third-party providers, operations managers, service managers, and users in the understanding of the end-to-end service system, as well as to enable governance at the service management and the service development levels.

These architecture frameworks have been defined through standards bodies and/or by private enterprises that recognize their advantage—standard processes that integrate the business-strategic processes and operations with the information technology and technology infrastructure (See Systems Engineering Standards). Most architecture frameworks model different scopes and levels of detail of business strategies, product and service offerings, business operations, and organizational aspects. Unfortunately, there are currently no frameworks that cover all the aspects (views) required to model the service systems. Some frameworks focus on business strategies, others in business process management, others in business operations, still others in aligning IT strategy or technology strategy to business strategy. Thus, a combination of architecture frameworks is required to create the enterprise service system model. For instance, an enterprise may use an enterprise business architecture (EBA) model covering strategic goals and objectives, business organization, and business services and processes where driven by market evolution, technology evolution, and customer demands. However, a reference framework would be needed to model the IT strategy (e.g., Information Technology Infrastructure Library (ITIL) v. 3 (OGC 2009)) and the organizations and processes needed to deliver, maintain, and manage the IT services according to the business strategy.

Service Architecture Frameworks

Prime examples of Service Architecture Frameworks are listed below.

Standards:

- Zachmann Framework (Zachman 2003)
- Business Process Modeling (BPM) (Hantry et al. 2010)
- The Open Group Architecture Framework (TOGAF) (TOGAF 2009)
- Enhanced-Telecomm Operations Map (eTOM) by the TeleManagemnt Forum (eTOM 2009)
- Service Oriented Architecture (SOA) (Erl 2008)
- National Institute of Standards and Technology (NIST) Smart Grid Reference Model (NIST 2010)
- Web services business process execution language (WS-BPEL) (OASIS 2007)
- Department of Defense Architecture Framework (DoDAF) (DoD 2010)
- Others.

Proprietary Enterprise Architecture Frameworks:

- Hewlett - Packard IT Service Management Reference Model (HP ITSMRM 2000)
- International Business Machines Systems Management Solutions Life Cycle, IBM Rational Software.
- Microsoft Operations Framework

This list represents only a sample of the existing service architecture frameworks.

One great example of architecture frameworks applications for service systems, the "High Level Reference Model for the Smart Grid," developed by NIST in 2010 under the "Energy Independence and Security Act of 2007" (EISA), is presented below:

> *EISA designated the development of a Smart Grid as a national policy goal, specifying that an interoperability framework should be "flexible, uniform and technology neutral. The law also instructed that the framework should accommodate "traditional, centralized generation and distribution resources" while also facilitating incorporation of new, innovative Smart Grid technologies, such as distributed renewable energy resources and energy storage.* (NIST 2010)

The NIST reference model was developed as "a tool for identifying the standards and protocols needed to ensure interoperability and cyber security, and defining and developing architectures for systems and subsystems within the smart grid." Figure 1 illustrates this model and the strategic (organizational), informational (business operations, data structures, and information exchanges required among system entities), and technical needs of the smart grid (data structures, entities specifications, interoperability requirements, etc.).

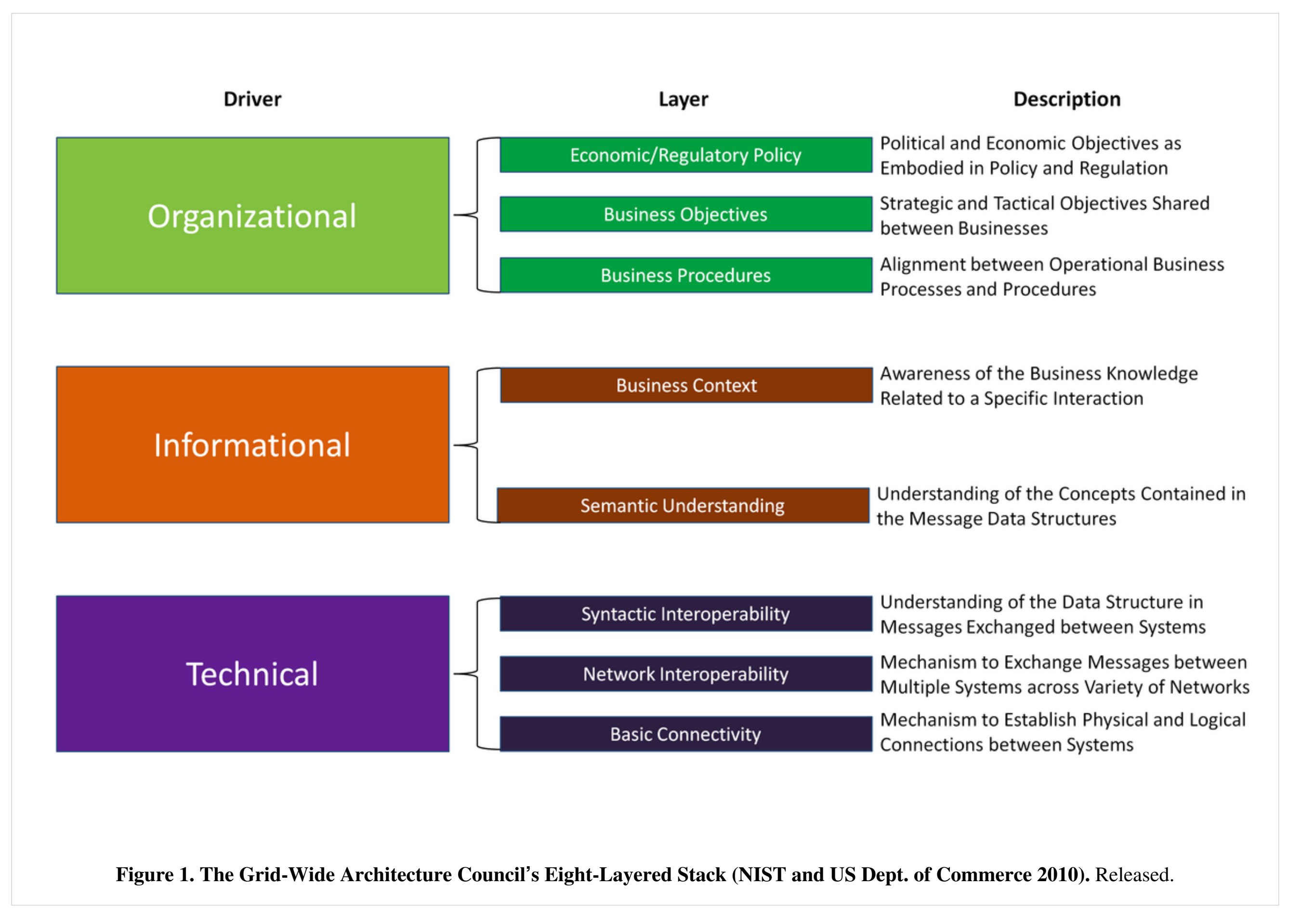


Figure 1. The Grid-Wide Architecture Council's Eight-Layered Stack (NIST and US Dept. of Commerce 2010). Released.

The NIST reference model uses this architecture framework to identify existing standards, identify new standards required for interoperability among interconnected networks, and to enable innovations where smart grid components (energy sources, bulk generation, storage, distribution, transmission, metering, cyber infrastructure, markets, service providers, customers, etc.) are supported by a broad range of interoperable options by well-defined interfaces useful across industries, including security. Emerging/innovative service development with massively scaled, well-managed, and secured networks will enable a dynamic market driven ecosystem representing new economic growth (NIST 2010).

This architecture framework is being used today by different standards organizations, such as the Smart Grid Interoperability Panel (SGIP), and several smart grid working groups. For details on priorities, working programs, and working group charters, see "High Level Reference Model for the Smart Grid" (NIST 2010).

For service systems, the application of any of these frameworks requires modifications/adaptations to create dynamic frameworks aware of environmental changes due to competitor's offerings, market demands, and customer co-creation. Most frameworks are static in nature; this requires business operations to manage changes through pre-defined (pre-programmed) processes for service configuration (glossary) and change control. Dynamic frameworks would allow real-time, or near real-time, analysis of impacts of newly discovered service on business processes, organizations, and revenue for run-time environment deployment.

Automatic service configuration and change control are being incorporated into the management process via service oriented architecture (SOA) for service automation (Gu et al. 2010) and service oriented computing (Maglio et al. 2010). In particular, progress has been made over the last ten years on the standards for adaptation, orchestration and creation of web services (WS) for service based applications (SBA). A good summary of existing life cycle approaches for adaptable and evolvable SBA is presented in (Papazoglou et al. 2010). Some examples of this are

- web services development life cycle (SDLC);

- rational unified process (RUP) for SOA;
- service oriented modeling and architecture (SOMA); and
- service oriented analysis and design/decision Modeling (SOAD).

Further research is required to understand the architectural implications of dynamic service configuration, including research on human behavior, social aspects, governance processes, business processes, and implications of dynamic service level agreements (SLA) for an enterprise service system. New ways are needed to include adaptation requirements for new technologies that will exchange information with the service system entities and may have their own specifications. These technologies include robots, sensors, renewable energy, nanotechnologies, three dimensional printers, and implantable medical devices.

References

Works Cited

Carroll, N., E. Whelan, and I. Richardson. 2010. "Applying Social Network Analysis to Discover Service Innovation within Agile Service Networks." *Service Science*. 2 (4): 225-244.

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence*. New York, NY, USA: John Wiley & Sons, Inc.

DoD. 2010. *DoD Architecture Framework (DoDAF)*, version 2.0. Arlington, VA, USA: US Department of Defense (DoD).

Erl, T. 2008. *SOA Principles of Service Design*. Boston, MA, USA: Prentice Hall, Pearson Education.

eTOM. 2009. "Business Process Framework." Morristown, NJ: TeleManagement Forum. Accessed May 30, 2011 at http://www.tmforum.org/BusinessProcessFramework/1647/home.html.

Grasso, D., and D. Martinelli. 2007. "Section B: Holistic Engineering." *The Chronicle Review, The Chronicles of Higher Education*. Vol. 53, Issue 28. Page 8B. March 2007.

Gu. Q., Cuadrado, F., Lago, P. and Duenãs, J.C. 2010. "Architecture views illustrating the service automation aspect of SOA". *Service research challenges and solutions for the future internet*. 339-372.

Hantry, F., M.P. Papazoglou, W. van den Heuvel, R. Haque, E. Whelan, N. Carroll, D. Karastoyanova, F. Leymann, C. Nikolaou, W. Lamersdorf, and M. Hacid. 2010. "Business Process Management," in *Service Research Challenges and Solutions for the Future Internet S-Cube – Towards Engineering, Managing and Adapting Service-Based Systems*, edited by M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger. Berlin and Heidelberg, Germany: Springer-Verlag. 27-54.

HP ITSMRM. 2000. "HP IT Service Management Reference Model. Technical White Paper." Palo Alto, California, USA: Hewlett – Packard Company. Accessed September 2, 2011. Available: ftp://ftp.hp.com/pub/services/itsm/info/itsm_rmwp.pdf.

Luzeaux, D., and J.R. Ruault (eds.). 2010. *Systems of Systems.* New York, NY, USA: John Wiley & Sons.

Maglio, P., and J. Spohrer. 2008. "Fundamentals of Service Science." *Journal of the Academy of Marketing Science.* 36 (1): 18-20.

National Institute of Standard and Technology (NIST). 2010. *NIST Framework and Roadmap for Smart Grid Interoperability Standards Release 1.0.* Gaithersburg, MD, USA: Office of the National Coordinator for Smart Grid Interoperability, US Department of Commerce. Accessed September 2, 2011. Available: http://www.nist.gov/smartgrid/upload/FinalSGDoc2010019-corr010411-2.pdf.

OASIS. 2007. "Web Services Business Process Execution Language Version 2.0." Organization for Advancement of Structured Information Standards (OASIS) Standard. Accessed September 2, 2011. Available: http://docs.oasis-open.org/webcgm/v2.0/OS/webcgm-v2.0.pdf.

OGC (Office of Government Commerce). 2009. *ITIL Lifecycle Publication Suite Books*. London, UK: The Stationery Office.

Papazoglou, M., K. Pohl, M. Parkin, and A. Metzger. 1998. "Service Research Challenges and Solutions for the Future Internet," in *Service Research Challenges and Solutions for the Future Internet S-Cube – Towards Engineering, Managing and Adapting Service-Based Systems*. Berlin and Heidelberg, Germany: Springer-Verlag.

TOGAF. 2009. "The Open Group Architecture Framework," version 9. The Open Architecture Group. Accessed September 2, 2011. Available: http://www.opengroup.org/togaf.

Zachman, J. 2003. "The Zachman Framework for Enterprise Architecture: Primer for Enterprise Engineering and Manufacturing." Accessed September 2, 2011. Available: http://www.zachmanframeworkassociates.com/index.php/ebook.

Primary References

Chang, C.M. 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* New York, NY, USA: John Wiley & Sons, Inc.

Erl, T. 2008. *SOA Principles of Service Design*. Boston, MA, USA: Prentice Hall, Pearson Education.

Hantry, F., M.P. Papazoglou, W. van den Heuvel, R. Haque, E. Whelan, N. Carroll, D. Karastoyanova, F. Leymann, C. Nikolaou, W. Lamersdorf, and M. Hacid. 2010. "Business Process Management," in *Service Research Challenges and Solutions for the Future Internet S-Cube – Towards Engineering, Managing and Adapting Service-Based Systems*, edited by M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger. Berlin and Heidelberg, Germany: Springer-Verlag. p. 27-54.

National Institute of Standard and Technology (NIST). 2010. *NIST Framework and Roadmap for Smart Grid Interoperability Standards Release 1.0.* Gaithersburg, MD, USA: Office of the National Coordinator for Smart Grid Interoperability, US Department of Commerce. Accessed September 2, 2011. Available: http://www.nist.gov/smartgrid/upload/FinalSGDoc2010019-corr010411-2.pdf.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Service Systems Engineering Stages

This article describes the stages of the service systems development process (SSDP) and expected outputs for each stage; for a closer alignment with the traditional systems engineering (TSE) process, the concept and feasibility phases have been combined into a single service strategy/concept as discussed in the SEBoK Systems Engineering and Management article. All of the stages of the SSDP take a similar iterative approach to fully understand the enterprise (glossary) capabilities, enterprise process impact, information technology (IT), and technology impacts and customer (glossary) expectations. Lin and Hsieh (2011) provide a good summary on New service Development processes. The Information Technology Infrastructure Library (ITIL) stage names have been purposely added to the SSDP to show the needed alignment between IT and technology. The reader should keep in mind that even though IT is crucial to the overall end-to-end system, service technology development needs must be taken into consideration in all the stages of SSDP.

Service Strategy/Concept

A service strategy/concept is the entry into the SSDP. The concept may be generated by an end-user (enterprise customer or consumer), a business manager, an engineering organization, new web service designers, new technology developments, and/or information technology trends. The service concept is the highest level of the service idea and it usually addresses what service is being proposed to what markets and to whom within these markets.

A high-level feasibility assessment of the concept is then carried out by the integrated service development team (ISDT) to assess the needs/impacts on enterprise process capabilities, operational capabilities, and/or new technology developments (access, infrastructure, operations support systems (OSS), service support systems (SSS), and business support systems (BSS). It should also consider any impacts on service governance, social, cultural, and human behaviors. The feasibility assessment also gives a plus or minus 30% estimate on the time to develop and the cost of development, which are entry points into the business case to evaluate whether the service is viable to develop and to market given the constraints and estimates. At this time, a decision (decision gate) determines if the service is to be developed.

If the business case is viable, then a detailed business description of the service is developed. This includes functions and features to be included, phases of development, markets to be addressed, customers within the markets to be targeted, and customer experiences expected from the service (i.e., defining the non-functional requirements of the service, such as the quality of service (QoS), availability, reliability, and security considerations and offerings within the service). This description allows detailed studies of expected human-computer interactions, social networking, technology requirements, and operations requirements. Governance and organizational process requirements should also be included to generate the “service description” as the main output from this stage.

Service systems engineering (SSE) takes an important role in understanding and eliciting the enterprise service concepts. Clearly, understood end-to-end business processes required for the intended service are fundamental to its successful development, deployment, and customer satisfaction. SSE works with business process management (BPM), social science, and cognitive science to elicit intended service operations, including target audiences, pre-sale, sale, and post-sale customer care processes.

Requirements Analysis and Engineering

A service requirements document is created that describes the service functions, the service entities, the intended interaction among entities, and the customer-facing and internal-facing functions/processes that are required to support the service. This description should conceptually include intended service level agreements (SLAs) and the obligations of the service provider process should there be any degree of non-compliance during service operation.

In addition to the TSE activities described earlier, the SSE requirements analysis and engineering process must develop a customer-centric view of the service to analyze SLA, QoS, value co-creation, monitoring, and assessment requirements to comply with the expected/planned SLA. This analysis will determine whether dynamic changes of the service are required during service operation to correct faults, reconfigure, administer, or to adapt/self-adapt for possible performance degradations.

Beyond the traditional service life cycle management (LCM) processes, the requirements must also be developed for service level management (SLM) processes and systems. These are needed to monitor, measure, and assess key performance indicators (KPIs), technical performance measures (TPMs), and service performance measures (SPMs) according to the SLA.

The SSE requirements analysis addresses the support systems for the governance, business, service, operations, and support processes to derive requirements for technologies, information systems, processes, and enterprise organizations. Interface requirements, information flows, and data requirements are also within the scope of requirements analysis. The main output is the service requirements document (SRD).

SSE plays a critical role in describing the services needs for day-to-day operations. These include customer care centers requirements and interfaces between network infrastructure provider(s), content provider(s), service provider(s), service based application provider(s), and the customer management process for the service. All of these are described in detail in the service operations plans (SOPs) and the operations technical plans (OTPs).

Systems Design/Development

The SRD, SOP, and OTP have enough detail regarding the service functions, operations, interfaces, and information flows required among the different service system entities to analyze, identify, and recommend end-to-end applicable architecture frameworks; to carry out trade-off analyses for the alternatives among service system entities; and to describe and allocate relationships (interactions) among entities at all levels of the service architecture. Detailed requirements are worked at lower levels to generate specifications for entity developers including data structures, data flow diagrams, and allocated performance requirements.

ITIL v. 3 (OGC 2007) recommends inclusion of the following service design processes:

- service catalog management,
- service level management,
- capacity management,
- availability management,
- service continuity management,
- security management, and
- supplier/provider management.

Service Integration, Verification & Validation

SSE defines integration and interface requirements for the seamless operation of the service. In this regard, the system engineer takes an integrator role to ensure proper data generation and flow through all the different systems composing the service offered. The goal is to ensure customers (consumer or internal) are getting the information required to carry out the tasks required in the business, operations, service, and customer processes. The service integration, verification, and validation plans need to include end-to-end verification and validation procedures for any new development or adaptations required for planned dynamic configuration/re-configuration of previously tested service systems. (See also System Verification and System Validation.)

The systems engineer creates these plans using a number of different perspectives. These include:

- end-to-end service (service validation test plans),
- customer care (operational readiness test plans),
- service provider (network validation test plans),
- service system entities interoperability/interface test plans,
- content provider (content validation test plans), and
- application (user acceptance test plans).

Service Transition/Deployment

Service systems may change very rapidly and new enhancements, new features, or new applications can be added as incremental developments, new developments, or adaptation to service offerings. Service systems engineers review new requirements to assess the feasibility of the changes to the service system entities, technologies, processes, and organizations, as well as their impacts on the service offerings. The service transition/deployment stage takes input from service development to plan for service insertion, technology insertion, processes adaptations, and implementation with minimal impact to existing services. During this stage, special care is taken with integration, verification, and validation test plans and regression testing to ensure new developments work flawlessly with existing services.

ITIL v. 3 (OGC 2007) recommends the following processes in the transition/deployment stage:

- transition planning and support,
- change management,
- service asset and configuration management,
- release and deployment management,
- service validation and testing,
- evaluation, and
- knowledge management.

Service Operations/Continuous Service Improvement (CSI)

Service operation manages the day-to-day activities of all aspects of the end-to-end service delivery to the customer. It manages the operations, administration, maintenance, and provisioning of the service, technology, and infrastructure required to deliver the contracted service to the customer within the specified service levels. The main service operations processes in ITIL v. 3 are

- event management,
- incident management,
- problem management,
- request fulfillment, and
- access management.

A continuous service improvement (CSI) plan for the implementation of technologies and tools for the continuous improvement of the service, monitoring, measuring, and analyzing process and service metrics is essential.

Service Systems Engineering Tools & Technologies

Tools and technologies from a broad spectrum of fields are extensively used during the different stages of SSE. Not only are they used for the development of the hardware, software, information systems and technology components, but also for the modelling, definition, and design of the organization, processes, and data structures of the service system (See also Representing Systems with Models). These tools and technologies include modelling, simulation, development, test bed, and social environmental aspects of the intended or to be designed service. The tools fall into three main domains:

1. business process management (BPM),
2. service design process, and
3. service design management.

Business process management (BPM) generally deals with process management scenarios to coordinate people and systems, including sequential workflow, straight through processing, case management, content life cycle management, collaborative process work, and value chain participation. Systems engineers work with service managers to align the business architectures with the technology and IT architecture. The business process modeling notation (BPMN) is a graphic notation standard that is implemented to describe a process's realization within any given workflow. This notation is linked with web services business process execution language (WS-BPEL), a format used to perform an automated business process by implementing web services technology. For an extensive review of existing BPM tools and BPM suites, please see Hantry et al. (2010), Carroll et al. (2010), Andrikoupolous et al. (2010), Lin and Hsieh (2011), and Ward-Dutton (2010).

Service design process: Architecture frameworks (AF) and enterprise architectures (EAs) are standards that help split complex systems (see also Complexity) into an interrelated, structured form. They describe the different characteristics of the products and services. Systems engineering modeling tools, such as the unified modeling language (UML) (OMG 2010a) and system modeling language (SysML) (OMG 2010b), help develop the AF and EA and greatly impact the continued evolution and successful implementation of complex projects. Service oriented architecture (SOA) and systems and software engineering architecture (ISO/IEC/IEEE 2011) are standards that apply architecture principles for specialized applications. Successful implementation of the architecture tools helps identify critical interfaces and improves understanding of the allocations between components and functions.

Mode-based systems engineering (MBSE), model driven architectures (MDA), and model oriented systems engineering (MOSES) are examples of commonly used tools for logical (functional), behavioral (operational), and physical design of the IT. UML, UML 2.0, and SysML are extensively used to describe operational scenarios, modes of operations, use cases, and entity relationships. For an extensive review of MBSE, MDA, and MOSES, please see Friedenthal (1998), Estefan (2008), Pezuela (2005), Andrikopoulos et al. (2010), and Hybertson (2010).

In addition, trade-off and engineering analyses use different optimization methodologies. Since services exhibit a significant level of randomness, statistical analysis, demand forecasting, multi-objective optimization, queuing theory, and stochastic optimization methodologies are tools used to model and simulate the service system behavior. These methodologies support decision making in areas as diverse as resource allocation, number of facilities, facilities' geographical locations, fleet routing and optimization, service systems reliability and prognosis, and network optimization. A good overview of these methodologies can be found in Daskin (2010).

During the service design process (SDP), planning for the implementation of technologies and tools for the continuous improvement of the service is performed. These tools support monitoring, measuring, and analyzing process and service performance metrics. The Deming cycle (plan, do, check, and act (PDCA) is widely used as the foundation for quality improvements across the service. Lean manufacturing, six sigma, swim lanes, balanced scoreboard, benchmarking, and gap analysis methodologies are commonly used for service evaluation and

continuous improvement.

Service design management: There are standards for implementing and managing systems engineering processes (IEEE 1220 (1998)) that help coordinate and synchronize all the service systems engineering processes leading to improved organizational collaboration and improved service delivery (see also Systems Engineering Standards). Standards have been developed in software engineering for product evaluation (ISO/IEC 14598 (1998)) and product quality (ISO/IEC 9126 series (2003a, 2003b, & 2004)), as well as information security management (ISO 27001 (2005)) and evaluation series (ISO 15408 (2008a, 2008b, & 2009)). The ITIL v. 3 describes best practices for IT service management, which can be extended to include service systems.

References

Works Cited

Adams, S., A. Cartlidge, A. Hanna, S. Rance, J. Sowerby, and J. Windebank. 2009. *ITIL V3 Foundation Handbook*. London, England, UK: The Stationary Office.

Andrikopoulos, V., A. Bucchiarone, E. Di Nitto, R. Kazhamiakin, S. Lane, V. Mazza, and I. Richardson. 2010. "Chapter 8: Service Engineering," in *Service Research Challenges and Solutions for the Future Internet S-Cube – Towards Engineering, Managing and Adapting Service-Based Systems*, edited by M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger. Berlin and Heidelberg, Germany: Springer-Verlag. p. 271-337.

Carroll, N., E. Whelan, and I. Richardson. 2010. "Applying Social Network Analysis to Discover Service Innovation within Agile Service Networks." *Service Science.* 2 (4): 225-244.

Daskin, M.S. 2010. *Service Science*. New York, NY, USA: John Wiley & Sons.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies*, rev B. Seattle, WA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Friedenthal, S. 1998. "Object Oriented System Engineering: Process Integration for 2000 and beyond." Presented at System Engineering & Software Symposium, Lockheed Martin Corporation, 1998, New Orleans, LA.

Hantry, F., M.P. Papazoglou, W. van den Heuvel, R. Haque, E. Whelan, N. Carroll, D. Karastoyanova, F. Leymann, C. Nikolaou, W. Lamersdorf, and M. Hacid. 2010. "Business Process Management," in *Service Research Challenges and Solutions for the Future Internet S-Cube – Towards Engineering, Managing and Adapting Service-Based Systems*, edited by M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger. Berlin and Heidelberg, Germany: Springer-Verlag. p. 27-54.

Hybertson, D.W. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boston, MA, USA: Auerbach Publications.

IEEE. 1998. IEEE 1220-1998, *IEEE Standard for Application and Management of the Systems Engineering Process*. Washington, DC, USA: Institute of Electrical and Electronics Engineers.

ISO/IEC. 1998. ISO/IEC 14598-5:1998, *Information technology — Part 5: Process for evaluators*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC. 2003a. ISO/IEC TR 9126-2:2003, *Software engineering — Product quality — Part 2: External metrics*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC. 2003b. ISO/IEC TR 9126-3:2003, *Software engineering — Product quality — Part 3: Internal metrics*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC. 2004. ISO/IEC TR 9126-4:2004, *Software engineering — Product quality — Part 4: Quality in use metrics*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC. 2005. ISO/IEC 27001:2005, *Information technology — Security techniques — Information security management systems — Requirements*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC. 2008a. ISO/IEC 15408-2:2008, *Information technology — Security techniques — Evaluation criteria for IT security — Part 2: Security functional components*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC. 2008b. ISO/IEC 15408-3:2008, *Information technology — Security techniques — Evaluation criteria for IT security — Part 3: Security assurance components*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC. 2009. ISO/IEC 15408-1:2009, *Information technology — Security techniques — Evaluation criteria for IT security* — Part 1: Introduction and general model. *Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.*

ISO/IEC/IEEE. 2011. ISO/IEC/IEEE 42010:2011, *Systems and software engineering — Architecture description*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

Lefever, B. 2005. "SeSCE Methodology." Rome, Italy: SeCSE Service Centric Systems Engineering. SeCSE511680. Available: http://www.secse-project.eu/wp-content/uploads/2007/08/a5_d4-secse-methodology-v1_3.pdf.

Lin, F., and P. Hsieh. 2011. "A SAT View on New Service Development." *Service Science.* 3 (2): 141-157.

OGC (Office of Government Commerce). 2007. *ITIL Lifecycle Publication Suite Books*. London, England, UK: The Stationery Office.

OMG. 2010a. *Unified Modeling Language™ (UML)*, version 2. Needham, MA, USA: Object Management Group. Available: http://www.omg.org/spec/UML/.

OMG. 2010b. *OMG Systems Modeling Language (SysML)*, version 1.2. Needham, MA, USA: Object Management Group. Available: http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf.

Pezuela, C. 2005. "Collection of Existing Service Centric Prototypes." Report A5.D1. Brussels, Belgium: European Union, Information Society Technology. Accessed September 5, 2011. Available: http://www.secse-project.eu/wp-content/uploads/2007/08/a5d1-collection-of-existing-service-centric-prototypes.pdf.

Ward-Dutton, N. 2010. "BPM Technology: Vendor Capability Comparison." MWD Premium Advisory Report. Horsham, West Sussex, UK: Macehiter Ward-Dutton (MWD) Limited. MWD Advisors. Accessed September 5, 2011. Available: http://www.mwdadvisors.com/library/detail.php?id=380.

Primary References

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies*, rev B. Seattle, WA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Hybertson, D.W. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boston, MA, USA: Auerbach Publications.

Lefever, B. 2005. "SeSCE Methodology." Rome, Italy: SeCSE Service Centric Systems Engineering. SeCSE511680. Available: http://www.secse-project.eu/wp-content/uploads/2007/08/a5_d4-secse-methodology-v1_3.pdf.

OGC (Office of Government Commerce). 2007. *ITIL Lifecycle Publication Suite Books*. London, England, UK: The Stationery Office.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Enterprise Systems Engineering

Enterprise Systems Engineering

Enterprise systems engineering (ESE) is the application of systems engineering principles, concepts, and methods to the planning, design, improvement, and operation of an enterprise.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Enterprise Systems Engineering Background
- The Enterprise as a System
- Related Business Activities
- Enterprise Systems Engineering Key Concepts
- Enterprise Systems Engineering Process Activities
- Enterprise Capability Management

Introduction

This knowledge area provides an introduction to systems engineering (SE) at the enterprise level in contrast to "traditional" SE (TSE) (sometimes called "conventional" or "classical" SE) performed in a development project or to "product" engineering (often called product development in the SE literature).

The concept of enterprise was instrumental in the great expansion of world trade in the 17th century (see note 1) and again during the Industrial Revolution of the 18th and 19th centuries. The world may be at the cusp of another global revolution enabled by the information age and the technologies and cultures of the Internet (see note 2). The discipline of SE now has the unique opportunity of providing the tools and methods for the next round of enterprise transformations.

> *Note 1. "The Dutch East India Company… was a chartered company established in 1602, when the States-General of the Netherlands granted it a 21-year monopoly to carry out colonial activities in Asia. It was the first multinational corporation in the world and the first company to issue stock. It was also arguably the* ***world's first mega-corporation****, possessing quasi-governmental powers, including the ability to wage war, negotiate treaties, coin money, and establish colonies." (emphasis added, National Library of the Netherlands 2010)*
>
> *Note 2. This new revolution is being enabled by cheap and easily usable technology, global availability of information and knowledge, and increased mobility and adaptability of human capital. The enterprise level of analysis is only feasible now because organizations can work together to form enterprises in a much more fluid manner.*

ESE is an emerging discipline that focuses on frameworks, tools, and problem-solving approaches for dealing with the inherent complexities of the enterprise. Furthermore, ESE addresses more than just solving problems; it also deals with the exploitation of opportunities for better ways to achieve the enterprise goals. A good overall description of ESE is provided by in the book by Rebovich and White (2011).

Key Terms

Enterprise

An enterprise consists of a purposeful combination (e.g., a network) of interdependent resources (e.g., people, processes, organizations, supporting technologies, and funding) that interact with

- each other to coordinate functions, share information, allocate funding, create workflows, and make decisions, etc.; and
- their environment(s) to achieve business and operational goals through a complex web of interactions distributed across geography and time (Rebovich and White 2011, 4-35).

The term enterprise has been defined as follows:

> (1) *One or more organizations sharing a definite mission, goals, and objectives to offer an output such as a product or service.* (ISO 2000);
>
> (2) *An organization (or cross organizational entity) supporting a defined business scope and mission that includes interdependent resources (people, organizations and technologies) that must coordinate their functions and share information in support of a common mission (or set of related missions).* (CIO Council 1999);
>
> (3) *The term enterprise can be defined in one of two ways. The first is when the entity being considered is tightly bounded and directed by a single executive function. The second is when organizational boundaries are less well defined and where there may be multiple owners in terms of direction of the resources being employed. The common factor is that both entities exist to achieve specified outcomes.* (MOD 2004); and
>
> (4) *A complex, (adaptive) socio-technical system that comprises interdependent resources of people, processes, information, and technology that must interact with each other and their environment in support of a common mission.* (Giachetti 2010)

An enterprise must do two things: (1) develop things within the enterprise to serve as either external offerings or as internal mechanisms to enable achievement of enterprise operations, and (2) transform the enterprise itself so that it can most effectively and efficiently perform its operations and survive in its competitive and constrained environment.

Enterprise vs Organization

It is worth noting that an enterprise is not equivalent to an "organization" according to the definition above. This is a frequent misuse of the term enterprise. The figure below shows that an enterprise includes not only the organizations that participate in it, but also people, knowledge, and other assets such as processes, principles, policies, practices, doctrine, theories, beliefs, facilities, land, intellectual property, and so on.

Some enterprises are organizations, but not all enterprises are organizations. Likewise, not all organizations are enterprises. Some enterprises have no readily identifiable "organizations" in them. Some enterprises are self-organizing (i.e., not organized by mandate) in that the sentient beings in the enterprise will find for themselves some way in which they can interact to produce greater results than can be done by the individuals alone. Self-organizing enterprises are often more flexible and agile than if they were organized from above (Dyer and Ericksen 2009; Stacey 2006).

> *One type of enterprise architecture that supports agility is a non-hierarchical organization without a single point of control. Individuals function autonomously, constantly interacting with each other to define the vision and aims, maintain a common understanding of requirements and monitor the work that needs to be done. Roles and responsibilities are not predetermined but rather emerge from individuals' self-organizing activities and are constantly in flux. Similarly, projects are generated*

everywhere in the enterprise, sometimes even from outside affiliates. Key decisions are made collaboratively, on the spot, and on the fly. Because of this, knowledge, power, and intelligence are spread through the enterprise, making it uniquely capable of quickly recovering and adapting to the loss of any key enterprise component. (http://en.wikipedia.org/wiki/Business_agility)

In spite of this lack of "organization" in some enterprises, SE can still contribute much in the engineering of the enterprise, as described in the articles below. However, SE must be prepared to apply some non-traditional approaches in doing so. Hence the need for embracing the new discipline called enterprise systems engineering (ESE).

Giachetti (2010) distinguishes between enterprise and organization by saying that an organization is a view of the enterprise. The organization view defines the structure and relationships of the organizational units, people, and other actors in an enterprise. Using this definition, we would say that all enterprises have some type of organization, whether formal, informal, hierarchical or self-organizing network.

Extended Enterprise

Sometimes it is prudent to consider a broader scope than merely the "boundaries" of the organizations involved in an enterprise. In some cases, it is necessary (and wise) to consider the "extended enterprise" in modeling, assessment, and decision making. This could include upstream suppliers, downstream consumers, and end user organizations, and perhaps even "sidestream" partners and key stakeholders. The extended enterprise can be defined as:

Wider organization representing all associated entities - customers, employees, suppliers, distributors, etc. - who directly or indirectly, formally or informally, collaborate in the design, development, production, and delivery of a product (or service) to the end user. (http://www.businessdictionary.com)

Enterprise Systems Engineering

Enterprise systems engineering (ESE), for the purpose of this article, is defined as the application of SE principles, concepts, and methods to the planning, design, improvement, and operation of an enterprise (see note 3). To enable more efficient and effective enterprise transformation, the enterprise needs to be looked at “as a system,” rather than merely as a collection of functions connected solely by information systems and shared facilities (Rouse 2009). While a systems perspective is required for dealing with the enterprise, this is rarely the task or responsibility of people who call themselves systems engineers.

Note 3. This form of systems engineering (i.e., ESE) includes (1) those traditional principles, concepts, and methods that work well in an enterprise environment, plus (2) an evolving set of newer ideas, precepts, and initiatives derived from complexity theory and the behavior of complex systems (such as those observed in nature and human languages).

Creating Value

The primary purpose of an enterprise is to create value for society, other stakeholders, and for the organizations that participate in that enterprise. This is illustrated in Figure 1 that shows all the key elements that contribute to this value creation process.

There are three types of organizations of interest: businesses, projects, and teams (see note 4). A typical business participates in multiple enterprises through its portfolio of projects. Large SE projects can be enterprises in their own right, with participation by many different businesses, and may be organized as a number of sub-projects.

Note 4. The use of the word “business” is not intended to mean only for-profit commercial ventures. As used here, it also includes government agencies and not-for-profit organizations, as well as commercial ventures. Business is the activity of providing goods and services involving financial, commercial, and industrial

aspects.

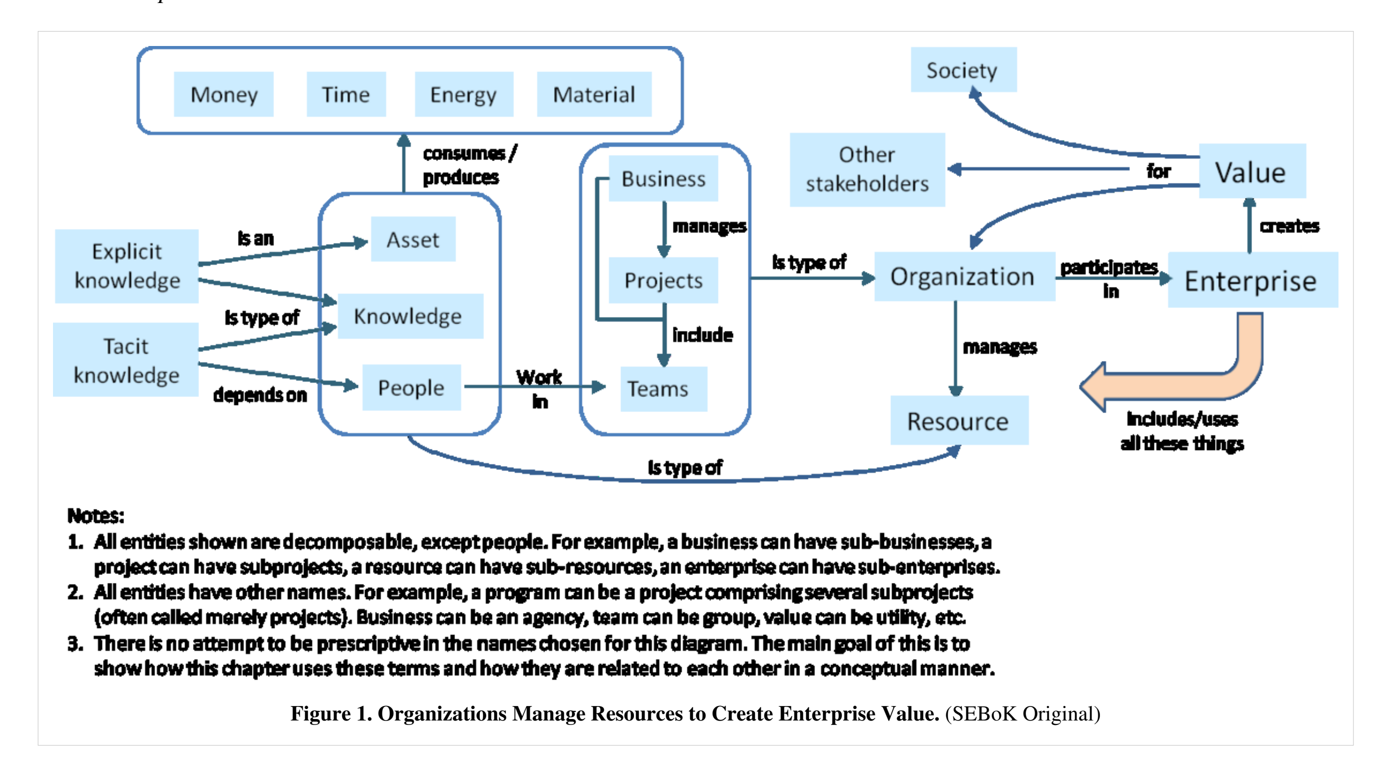


Figure 1. Organizations Manage Resources to Create Enterprise Value. (SEBoK Original)

Resource Optimization

A key choice for businesses that conduct SE is to what extent, if at all, they seek to optimize their use of resources (people, knowledge, assets) across teams, projects, and business units. Optimization of resources is not the goal in itself, but rather a means to achieve the goal of maximizing value for the enterprise and its stakeholders. At one extreme, in a product-oriented organization, projects may be responsible for hiring, training, and firing their own staff, as well as managing all assets required for their delivery of products or services. (The term "product-oriented organization" is not meant in the sense of product-oriented SE, but rather in the sense of this being one of the basic constructs available when formulating organizational strategy.)

At the other extreme, in a functional organization, the projects delegate almost all their work to functional groups. In between these two extremes is a matrix organization that is used to give functional specialists a “home” between project assignments. A full discussion of organizational approaches and situations along with their applicability in enabling SE for the organization is provided in the article called Systems Engineering Organizational Strategy.

The optimization debate can be handled as described in the book called "Enterprise Architecture as Strategy" (Ross, Weill, and Robertson 2006). In other words, an enterprise can choose (or not) to unify its operations and can choose (or not) to unify its information base. There are different strategies the enterprise might adopt to achieve and sustain value creation (and how ESE helps an enterprise to choose). This is further addressed in the section on Enterprise Architecture Formulation & Assessment in the article called Enterprise Capability Management.

Enabling Systems Engineering in the Organization

SE skills, techniques, and resources are relevant to many enterprise functions, and a well-founded SE capability can make a substantial contribution at the enterprise level, as well as at the project level. The article called Systems Engineering Organizational Strategy discusses enabling SE in the organization, while the article called Enabling Businesses and Enterprises focuses on the cross-organizational functions at the business and enterprise levels. The competence of individuals is discussed in the article called Enabling Individuals.

Kinds of Knowledge Used by the Enterprise

Knowledge is a key resource for ESE. There are generally two kinds of knowledge: explicit and tacit. Explicit knowledge can be written down or incorporated in computer codes. Much of the relevant knowledge, however, is "tacit knowledge" that only exists within the heads of people and in the context of relationships that people form with each other (e.g., team, project, and business level knowledge). The ability of an organization to create value is critically dependent on the people it employs, on what they know, how they work together, and how well they are organized and motivated to contribute to the organization's purpose.

Projects, Programs & Businesses

The term "program" is used in various ways in different domains. In some domains a team can be called a program (e.g., a customer support team is their customer relationship "program"). In others, an entire business is called a program (e.g., a wireless communications business unit program), and in others the whole enterprise is called a program (e.g., the Joint Strike Fighter program and the Apollo Space program). And in many cases, the terms project and program are used interchangeably with no discernible distinction in their meaning or scope. Typically, but not always, there are program managers who have profit and loss (P&L) responsibility and are the ultimate program decision makers. A program manager may have a portfolio of items (services, products, facilities, intellectual property, etc.) that are usually provided, implemented, or acquired through projects.

The Office of Government Commerce provides a useful distinction between programs and projects:

> *The ultimate goal of a Programme is to realise outcomes and benefits of strategic relevance. To achieve this a programme is designed as a temporary flexible organisation structure created to coordinate, direct and oversee the implementation of a set of related projects and activities in order to deliver outcomes and benefits related to the organisation's strategic objectives...*
>
> *A programme is likely to have a life that spans several years. A Project is usually of shorter duration (a few months perhaps) and will be focussed on the creation of a set of deliverables within agreed cost, time and quality parameters.* (OGC 2010)

Practical Considerations

When it comes to performing SE at the enterprise level, there are several good practices to keep in mind (Rebovich and White 2011):

- Set enterprise fitness as the key measure of system success. Leverage game theory and ecology, along with the practices of satisfying and governing the commons.
- Deal with uncertainty and conflict in the enterprise through adaptation: variety, selection, exploration, and experimentation.
- Leverage the practice of layered architectures with loose couplers and the theory of order and chaos in networks.

Enterprise governance involves shaping the political, operational, economic, and technical (POET) landscape. One should not try to control the enterprise like one would in a TSE effort at the project level.

References

Works Cited

BusinessDictionary.com, "Extended Enterprise." Accessed September 12, 2012. Available: http:/ / www. businessdictionary.com/definition/extended-enterprise.html.

CIO Council. 1999. *Federal Enterprise Architecture Framework (FEAF)*. Washington, DC, USA: Chief Information Officer (CIO) Council.

Dyer, L., and J. Ericksen. 2009. "Complexity-based Agile Enterprises: Putting Self-Organizing Emergence to Work," in *The Sage Handbook of Human Resource Management,* edited by A. Wilkinson et al. London, UK: Sage. p. 436–457.

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

ISO. 2000. ISO 15704:2000, *Industrial Automation Systems -- Requirements for Enterprise-Reference Architectures and Methodologies*. Geneva, Switzerland: International Organization for Standardization (ISO).

MOD. 2004. *Ministry of Defence Architecture Framework (MODAF)*, version 2. London, UK: UK Ministry of Defence.

National Library of the the Netherlands. 2010. "Dossier VOC (1602-1799)." Accessed September 12, 2012. Available: http://www.kb.nl/dossiers/voc/voc.html (in Dutch).

OGC (Office of Government Commerce). 2010. *Guidelines for Managing Programmes: Understanding programmes and programme management*. London, UK: The Stationery Office.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, Auerbach.

Ross, J.W., P. Weill, and D. Robertson. 2006. *Enterprise Architecture As Strategy: Creating a Foundation for Business Execution*. Boston, MA, USA: Harvard Business Review Press.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of Systems Engineering and Management,* 2nd ed., edited by A.P. Sage and W. B. Rouse. New York, NY, USA: Wiley and Sons, Inc.

Stacey, R. 2006. "The Science of Complexity: An Alternative Perspective for Strategic Change Processes," in *Complexity and Organization: Readings and Conversations*, edited by R. MacIntosh et al. London, UK: Routledge. p. 74–100.

Wikipedia contributors, "Business agility," *Wikipedia, The Free Encyclopedia*. Accessed November 28, 2012. Available: http://en.wikipedia.org/w/index.php?title=Business_agility&oldid=503858042.

Primary References

Bernus, P., L. Nemes, and G. Schmidt (eds.). 2003. *Handbook on Enterprise Architecture*. Berlin and Heidelberg, Germany: Springer-Verlag.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, Auerbach.

Rouse, W.B. 2005. "Enterprise as Systems: Essential Challenges and Enterprise Transformation." *Systems Engineering, the Journal of the International Council on Systems Engineering (INCOSE)*. 8 (2): 138-50.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of Systems Engineering and Management*, 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY, USA: Wiley and Sons, Inc.

Valerdi, R. and D.J. Nightingale. 2011. "An Introduction to the Journal of Enterprise Transformation." *Journal of Enterprise Transformation.* 1 (1): 1-6.

Additional References

Drucker, P.F. 1994. "The theory of business." *Harvard Business Review*. 72 (5): 95-104.

Fox, M., J.F. Chionglo, and F.G. Fadel. 1993. "A common sense model of the enterprise." Presented at the 3rd Industrial Engineering Research Conference, 1993, Norcross, GA, USA.

Joannou, P. 2007. "Enterprise, systems, and software—the need for integration." *Computer*. 40 (5): 103-105.

Journal of Enterprise Architecture. Available: http://www.globalaea.org/?page=JEAOverview.

MITRE. 2012. "Enterprise Engineering," in *Systems Engineering Guide*, MITRE Corporation. Accessed 8 July 2012. Available: http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/.

Nightingale, D., and J. Srinivasan. 2011. *Beyond the Lean Revolution: Achieving Successful and Sustainable Enterprise Transformation*. New York, NY, USA: AMACOM Press.

Nightingale, D., and R. Valerdi (eds). *Journal of Enterprise Transformation*. London, UK: Taylor & Francis. Available: http://www.tandf.co.uk/journals/UJET.

Saenz, O.A. 2005. "Framework for Enterprise Systems Engineering," in *FIU Electronic Theses and Dissertations*. Miami, FL, USA: Florida International University. Accessed September 12, 2012. Available: http:/ / digitalcommons.fiu.edu/cgi/viewcontent.cgi?article=1055&context=etd.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Enterprise Systems Engineering Background

This article provides a common context for the succeeding topics in the knowledge area.

Capabilities in the Enterprise

The enterprise acquires or develops systems or individual elements of a system. The enterprise can also create, supply, use, and operate systems or system elements. Since there could possibly be several organizations involved in this enterprise venture, each organization could be responsible for particular systems or perhaps for certain kinds of elements. Each organization brings their own organizational capability with them and the unique combination of these organizations leads to the overall operational capability of the whole enterprise. These concepts are illustrated below.

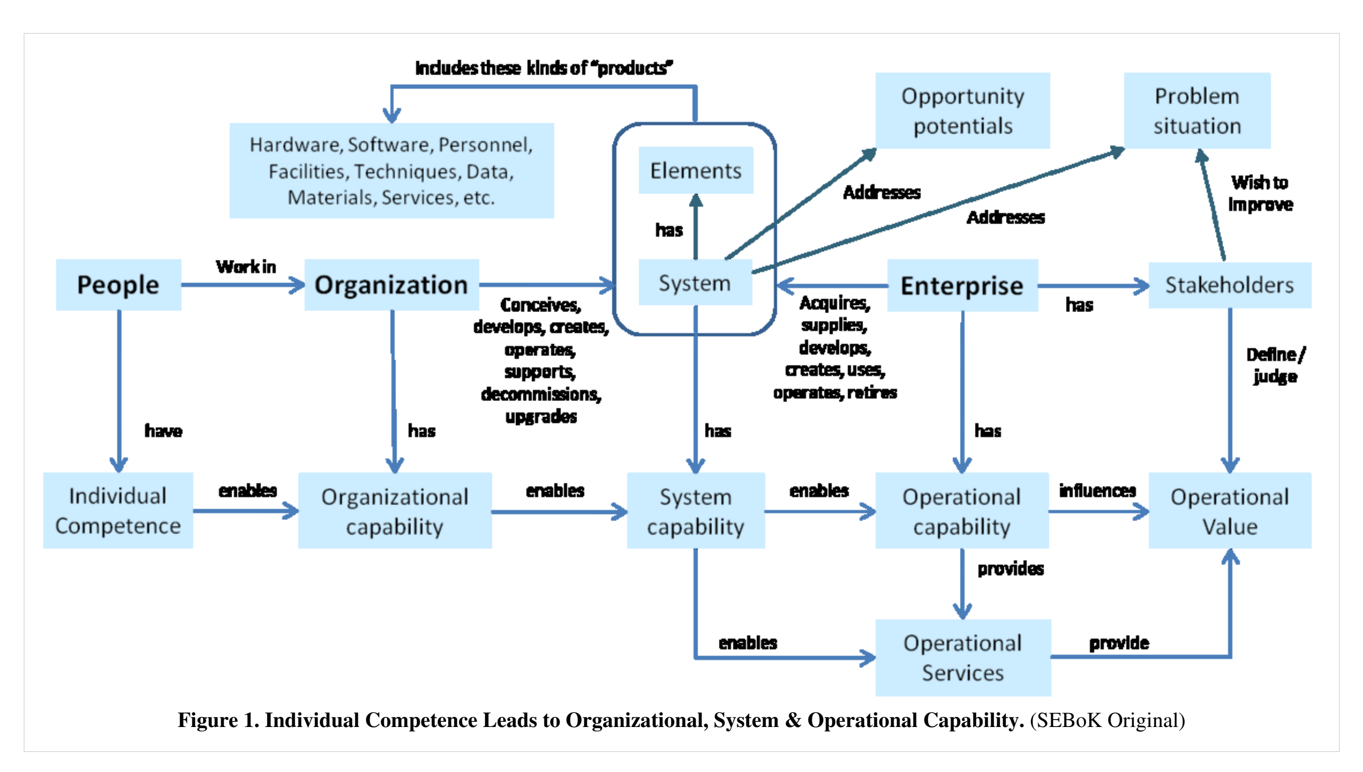


Figure 1. Individual Competence Leads to Organizational, System & Operational Capability. (SEBoK Original)

Organizational capabilities are addressed in the article on Systems Engineering Organizational Strategy, and individual competencies are addressed in the article on Enabling Individuals as they relate to the principles, theories, and practices of organizational behavior.

Organizational Capabilities and Competencies

The word "capability" is used in systems engineering (SE) in the sense of "the ability to do something useful under a particular set of conditions." This article discusses three different kinds of capabilities: organizational capability, system capability, and operational capability. It uses the word "competence" to refer to the ability of people relative to the SE task. Individual competence, (sometimes called "competency"), contributes to, but is not the sole determinant of, organizational capability. This competence is translated to organizational capabilities through the work practices that are adopted by the organizations. New systems (with new or enhanced system capabilities) are developed to enhance enterprise operational capability in response to stakeholder's concerns about a problem situation.

Enterprise stakeholders are the ultimate arbiters of value (glossary) for the system to be delivered. Organizational, system, and operational capabilities cannot be designed, improved, and implemented independently. The key to understanding the dependencies between capabilities is through architecture modeling and analysis as part of the activities described in the article called Enterprise Capability Management. "Capability engineering" is an emerging discipline that could enhance the effectiveness of enterprise systems engineering (ESE), which is further discussed in the article on Systems of Systems (SoS).

Organizational Design

The competencies of individuals are important to the overall organizational capability as discussed in the article on Enabling Individuals. The organizational capability is also a function of how the people, teams, projects, and businesses are organized. The organizational design should specify the roles, authorities, responsibilities, and accountabilities (RARA) of the organizational units to ensure the most efficient and effective operations. Effectiveness of enterprise operations is certainly driven by management principles, concepts, and approaches, but it is also largely driven by its leadership principles, concepts, and approaches. These factors are discussed in the article on Systems Engineering Organizational Strategy that discusses how to organize for effective performance of SE.

Organizational structure is tightly tied to creating value for the enterprise's various stakeholders. Since the enterprise is made up of various elements including people, processes, technologies, and assets, the organizational structure of the people and the allocation of responsibilities for executing portions of the value stream is a "design decision" for the enterprise and hence is a key element of properly performing ESE. Organizational design is increasingly influenced by the portfolio of products and services and the degree of coupling between them. This organizational design will be based on organizational design patterns and their tradeoffs, as discussed in the article on Systems Engineering Organizational Strategy. Browning (2009) discusses one approach for modeling and analysis of an organization.

Operational Capabilities & Operational Services

As you can see in this figure, operational capabilities provide operational services that are enabled by system capabilities. These system capabilities are inherent in the system that is conceived, developed, created and/or operated by an enterprise. ESE concentrates its efforts on maximizing operational value for various stakeholders, some of whom may be interested in the improvement of some problem situation.

ESE, however, addresses more than just solving problems; it also deals with the exploitation of opportunities for better ways to achieve the enterprise goals. This opportunity might involve lowering of operating costs, increasing market share, decreasing deployment risk, reducing time to market, and any number of other enterprise goals. The importance of addressing opportunity potentials should not be underestimated in the execution of ESE practices.

This article focuses on the *operational capabilities* of an enterprise and the contribution of these capabilities to *operational value* (as perceived by the stakeholders). Notice that the organization or enterprise can deal with either the system as a whole or with only one (or a few) of its elements. These elements are not necessarily hard items, like hardware and software, but can also include "soft" items, like people, processes, principles, policies, practices, organizations, doctrine, theories, beliefs, and so on.

Services vs. Products vs. Enterprises

A service system is a collection of items (or entities) that perform the operations, administration, management and provisioning (OAM&P) of resources that together provide the opportunities to co-create value (glossary) by both the service provider and the service consumer.

A collection of services is not necessarily a service system. In fact, this collection of services is often merely a product system that is one of the resources being OAM&P'ed by the service system. A product system can be composed of hardware, software, personnel (see note 1), facilities, data, materials, techniques, and even services. Each of these product system elements can be "engineered."

> *Note 1. Even personnel are engineered in the sense that their roles and responsibilities are specified precisely and trade-offs are made about which functions are performed by these people versus by hardware or software. People are "produced" in the sense that untrained people are trained to perform their allocated system functions, unknowledgeable people are educated to find or create the information they need to do their assigned task, and uninformed people are taught how to get access to the data they need, and how to extract relevant information from that data.*

It is important to understand the difference between the services "enabled" by a service system versus the services that are the elements of a service system entity. See the Service Systems Engineering article for more information about services and how they are engineered.

Likewise, a collection of services is not necessarily an enterprise system. An enterprise may be composed of service systems, along with product systems, as well as policies, procedures, properties, knowledge, financial capital, intellectual capital, and so on. An enterprise might even contain sub-enterprises. Enterprise SE must do the engineering not only across the enterprise itself, but may also get involved in the engineering of the service systems and products systems that the enterprise depends on in order to achieve its goals.

Enterprise Components

The above depictions of enterprise-related things do not show the components of an enterprise. The components of an enterprise when it is viewed as a “system” are different than the components of a product or service system (which is the focus of most literature on systems engineering). The figure below shows the typical kinds of components (shown here as “domains”) in an enterprise (Troux 2010) that could be utilized in achieving the desired enterprise *operational capability* as shown in Figure 1. It is this operational capability that drives ultimate value for the enterprise’s customers and other stakeholders. Further discussion on enterprise components is provided by Reese (2010) and Lawson (2010, chap. 8).

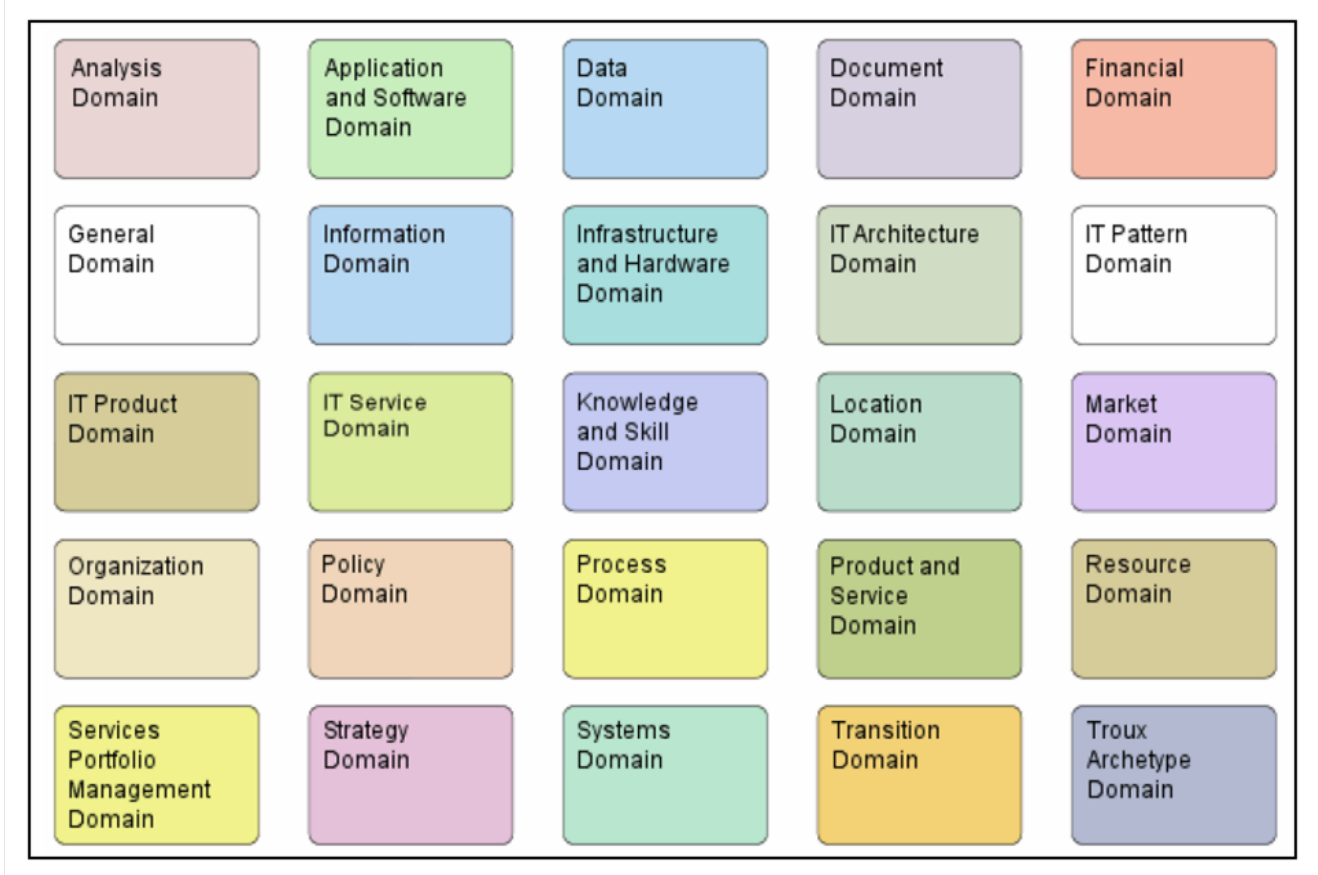


Figure 2. Categories of Enterprise Components (Troux Technologies, 2010). Reprinted with permission of Copyright © 2010 Troux Technologies. All other rights are reserved by the copyright owner.

The application/software and infrastructure/hardware domains (shown above) are likely the most familiar to systems engineers. The application/software domain contains things like the deployed software itself plus applications, modules, servers, patches, functions, and messages. The infrastructure/hardware domain contains things like the hardware itself plus networks and different kinds of hardware like computing hardware, cabinets, and network devices. There might different subtypes of computing hardware like computers, servers, desktops, laptops, and mainframes.

This particular "semantic model" had its origins in the area of information technology (IT) management but has been successfully expanded beyond the IT domain (Martin 2003 and 2005). You can see from this elaboration of these domains that an enterprise architecture "schema" can be quite extensive in the kinds of things it can model. The less technical domains would be things like policy, market, strategy, transition, financial, knowledge and skill, and analysis. In a typical enterprise architecture schema like this there could be over a hundred types of modeling objects grouped into these domains.

Various tools used in modeling the enterprise are described at http://www.enterprise-architecture.info/EA_Tools.htm (IEAD 2011). The TOGAF metamodel (http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap34.

html) used in The Open Group Architecture Framework (TOGAF) is another useful depiction of the various modeling entities involved in modeling the enterprise (TOGAF 2009).

Scope of Enterprise SE

Computer and communications technologies make it easier to integrate activities across the enterprise, but this does not necessarily make the enterprise more effective and efficient. To enable this to happen, one needs to look at the whole enterprise as a system, rather than as a collection of functions connected solely by information systems and shared facilities.

Essential Challenges

Enterprises face strategic challenges that are essential to address in order to ensure that the enterprise will succeed (Rouse 2009):

- **Growth:** Increasing impact, perhaps in saturated/declining "markets",
- **Value:** Enhancing relationships of processes to benefits and costs,
- **Focus:** Pursuing opportunities and avoiding diversions,
- **Change:** Competing creatively while maintaining continuity,
- **Future:** Investing in inherently unpredictable outcomes,
- **Knowledge:** Transforming information to insights to programs, and
- **Time:** Carefully allocating the organization's scarcest resource.

To address these challenges, one recognizes that the central source of value in the enterprise is in its people. "Understanding and supporting the interests of an enterprise's diverse stakeholders — and finding the 'sweet spot' among the many competing interests — is a central aspect of discerning the work of the enterprise as a system and creating mechanisms to enhance this work" (Rouse 2009).

Enterprise Transformation

Enterprises are constantly transforming, whether at the individual level (wherein individuals alter their work practices) or at the enterprise level (large-scale planned strategic changes) (Srinivasan 2010). These changes are a response on the part of the enterprise to evolving opportunities and emerging threats. It is not merely a matter of doing work better, but doing different work, which is often a more important result. Value is created through the execution of business processes. However, not all processes necessarily contribute to overall value (Rouse 2005, 138-150). It is important to focus on process and how they contribute to the overall value stream.

After gaining a good understanding of business processes, the next main concern is how best to deploy and manage the enterprise's human, financial, and physical assets. The key challenge in transforming an enterprise is, in the midst of all this change, continuing to *satisfice* key stakeholders (see note 2).

> *Note 2. "Satisfice" means to decide on and pursue a course of action satisfying the minimum requirements to achieve a goal. For the enterprise as a whole, it is often impossible to completely satisfy all stakeholders given their competing and conflicting concerns and interests. Therefore, the concept of "satisficing" is a very important element in the execution of ESE practices. It has less stringent criteria than the concept of "satisfaction," which is commonly used in product/service systems engineering.*

Systems engineers have to respond to an increased recognition of the 'connectedness' of products and systems, brought about by a number of trends, for example: the capability of (mainly digital) technology, working across multiple systems, to transform businesses and operational systems; the need to create systems in families to increase product diversity and reuse technology, in order to reduce development and operating costs; and the need to build systems which can be brought together flexibly in operations, even if such co-operation was not foreseen at the time of development.

There has also been an increase in collaborative systems development activities, often spanning national boundaries. This has proceeded alongside a growth in the development of what might be called *meta-systems,* that is systems comprising parts which would previously have been considered as complex in their own right a generation ago, now conceived of and developed as a whole, and thus requiring fresh approaches, of the adaption of old ones.

Tackling these issues requires an approach that transcends the technical and process domain. ESE needs to address integration at the organizational and value chain level.

Transformation Context

Enterprise transformation occurs in the external context of the economy and markets as shown in the figure below (Rouse 2009). The “market” for the enterprise can be thought of as the context in which the enterprise operates. Of course, in the public sector, the enterprise’s “market” is commonly known as its “constituency.”

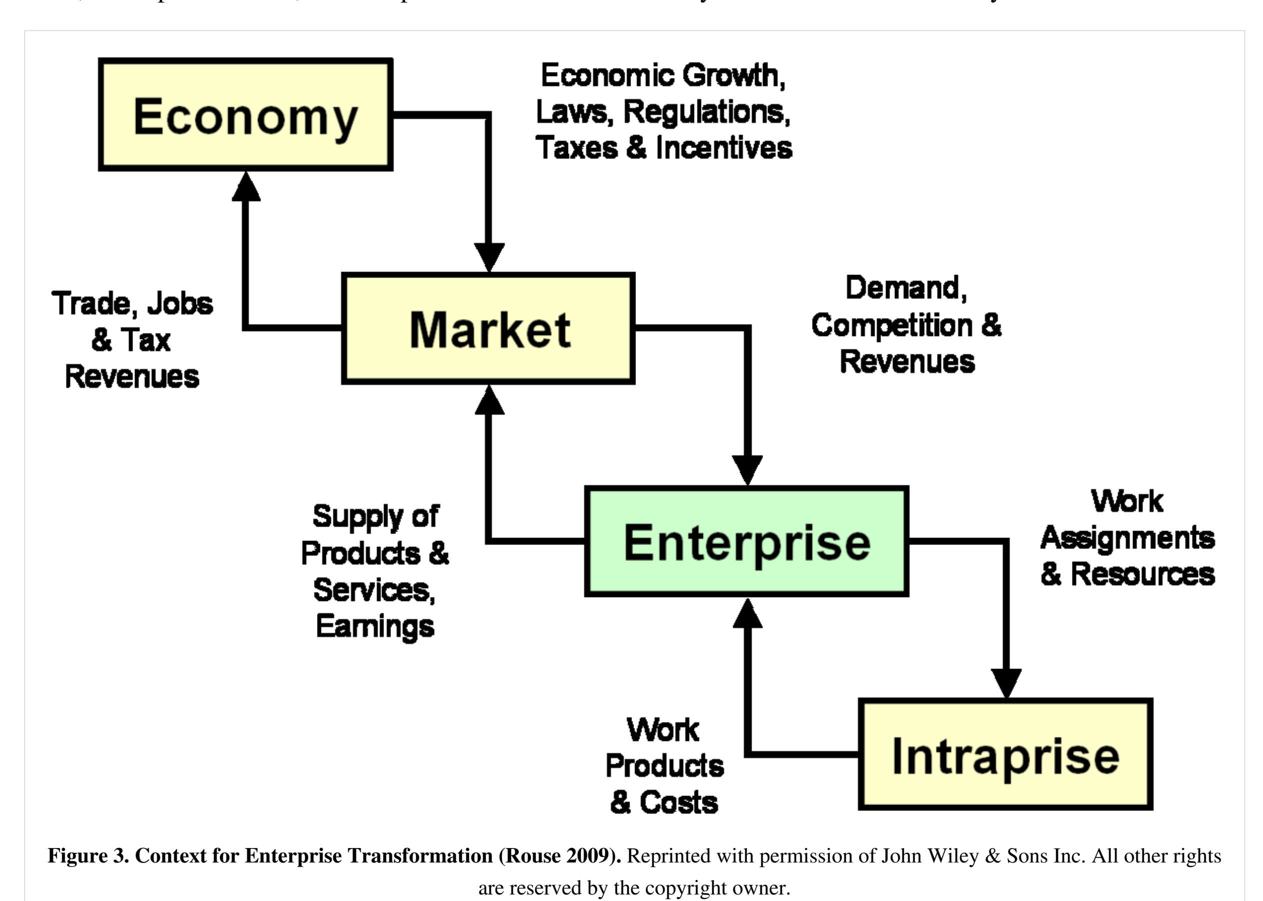


Figure 3. Context for Enterprise Transformation (Rouse 2009). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

The term “intraprise” is used here to denote the many systems internal to the enterprise. This includes "information systems such as... ERP [enterprise resource planning] systems, as well as social and cultural systems. More specifically, work assignments are pursued via work processes and yield work products, incurring costs" (Rouse 2009). The social and cultural aspects of an enterprise are addressed further in the article called Enabling Businesses and Enterprises.

Modeling the Enterprise

Models of the enterprise can serve as the basis for understanding the enterprise in its context of markets and economies. The figure below shows the various drivers (or inputs) of an enterprise and its potential outcomes (or outputs) (Rouse 2009). Enterprise architecture can be a key enabler for modeling and can serve as a basis for transformation (Vernadat 1996; Bernus, Laszlo, and Schmidt 2003; Nightingale and Rhodes 2004). Enterprise architecture can be used to provide a model to understand how the parts of the enterprise fit together (or do not)

(Giachetti 2010) (See also Representing Systems with Models). For a good review of the subject see Lillehagen and Krogstie (2008).

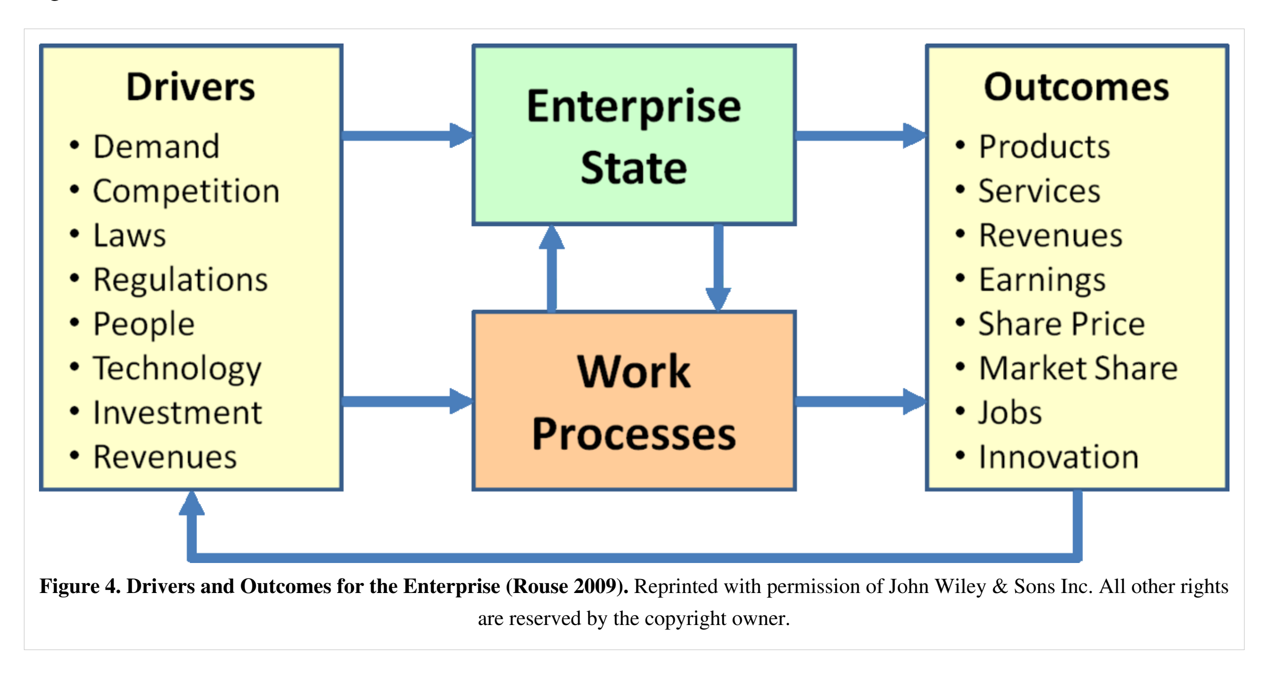


Figure 4. Drivers and Outcomes for the Enterprise (Rouse 2009). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

In Pursuit of Value

Based on his theory of enterprise transformation, Rouse (2005, 279-295) has identified four alternative perspectives that tend to drive the need for transformation:

1. **Value Opportunities:** The lure of greater success via market and/or technology opportunities prompts transformation initiatives.
2. **Value Threats:** The danger of anticipated failure due to market and/or technology threats prompts transformation initiatives.
3. **Value Competition:** Other players' transformation initiatives prompt recognition that transformation is necessary to continued success.
4. **Value Crises:** Steadily declining market performance, cash flow problems, etc., prompt recognition that transformation is necessary for the enterprise to survive.

Work processes can be enhanced, streamlined, eliminated, and invented to help in the pursuit of enhanced value. These process changes should be aligned with enterprise strategy to maximize value produced by the enterprise (Hammer and Champy 1993). As shown below, there are many entities involved in helping the enterprise create value for society, participating organizations, and other stakeholders.

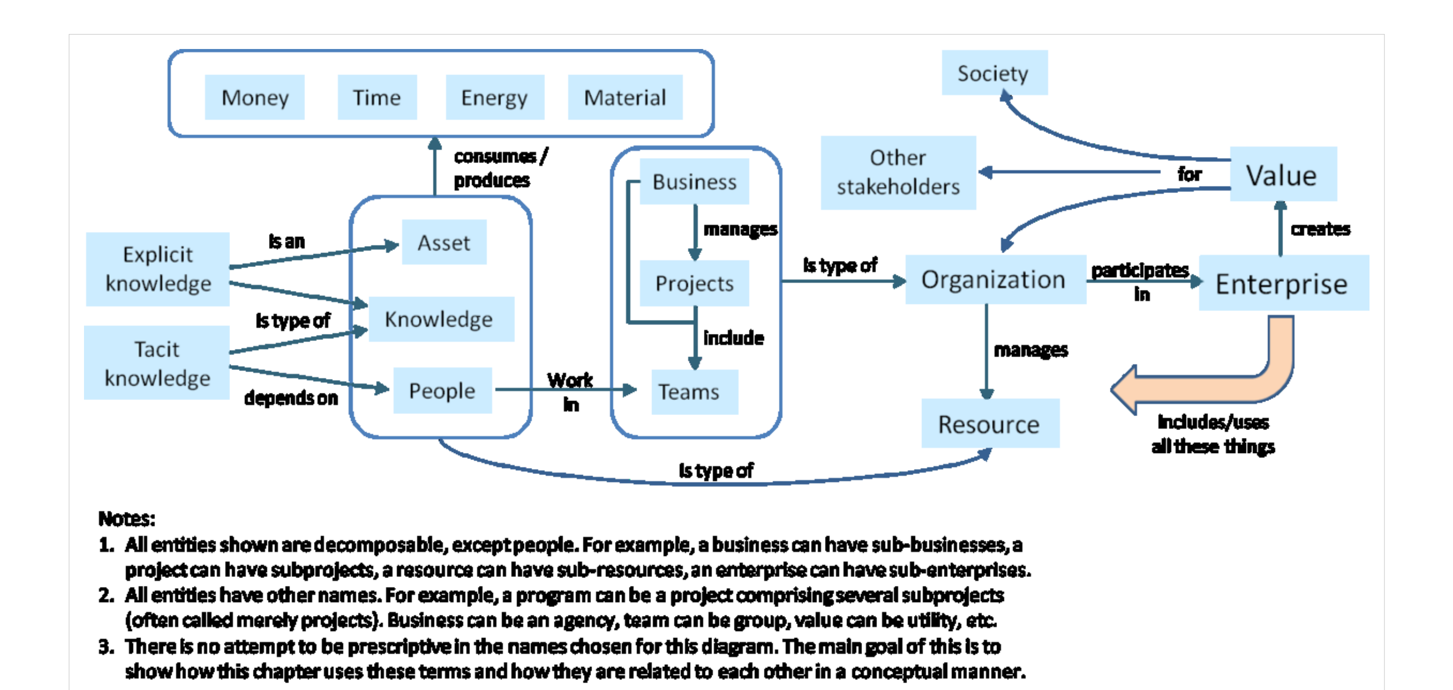


Figure 5. Organizations Manage Resources to Create Enterprise Value. (SEBoK Original)

References

Works Cited

Browning, T.R. 2009. "Using the Design Structure Matrix to Design Program Organizations," in *Handbook of Systems Engineering and Management*, 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY: Wiley and Sons, Inc.

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group.

Hammer, M., and J. Champy. 1993. *Reengineering the Corporation: A Manifesto for Business Revolution*. New York, NY: Harper Business, HarperCollins Publishers.

IEAD. 2011. "Enterprise Architecture Tools." Institute for Enterprise Architecture Developments. Accessed September 12, 2012. Available: http://www.enterprise-architecture.info/EA_Tools.htm.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. Kings College, UK: College Publications.

Lillehagen, F., and J. Krogstie. 2008. "Chapter 4: State of the Art of Enterprise Modelling," in *Active Knowledge Management of Enterprises*. New York, NY, USA: Springer.

Martin, J.N. 2003. "On the Use of Knowledge Modeling Tools and Techniques to Characterize the NOAA Observing System Architecture." Presented at 13th Annual International Council on Systems Engineering (INCOSE) International Symposium, 2003, Arlington, VA, USA.

Martin, J.N. 2005. "Using an Enterprise Architecture to Assess the Societal Benefits of Earth Science Research." Presented at 15th Annual International Council on Systems Engineering (INCOSE) International Symposium, 2005, Rochester, NY, USA.

Miller, J., and S. Page. 2007. *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*. Princeton, NJ, USA: Princeton University Press.

Reese, R.J. 2010. *Troux Enterprise Architecture Solutions*. Birmingham, UK: Packt Publishing Ltd.

Rouse, W.B. 2005. "Enterprise as Systems: Essential Challenges and Enterprise Transformation." *Systems Engineering, the Journal of the International Council on Systems Engineering (INCOSE)*. 8 (2): 138-150.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of Systems Engineering and Management*, 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY: Wiley and Sons, Inc.

Srinivasan, J. 2010. "Towards a Theory Sensitive Approach to Planning Enterprise Transformation." Presented at 5th European Institute for Advanced Studies in Management (EIASM) Workshop on Organizational Change and Development, September 23-24, 2010, Vienna, Austria.

TOGAF 2009. "The Open Group Architecture Framework," version 9. The Open Architecture Group. Accessed September 2, 2011. Available: http://www.opengroup.org/togaf.

Troux. 2010. *Metamodeling and modeling with Troux Semantics,* version 9. Austin, TX, USA: Troux Technologies.

White, B.E. 2009. "Complex Adaptive Systems Engineering (CASE)." Presented at IEEE Systems Conference, March 23-26, 2009, Vancouver, Canada.

Primary References

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of Systems Engineering and Management*, 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY: Wiley and Sons, Inc.

Srinivasan, J. 2010. "Towards a Theory Sensitive Approach to Planning Enterprise Transformation." Presented at 5th European Institute for Advanced Studies in Management (EIASM) Workshop on Organizational Change and Development, September 23-24, 2010, Vienna, Austria.

White, B.E. 2009. "Complex Adaptive Systems Engineering (CASE)." Presented at IEEE Systems Conference, March 23-26, 2009, Vancouver, Canada.

Additional References

McCarter, B.G., and B.E. White. 2009. "Emergence of SoS, sociocognitive aspects," in *Systems of systems engineering: Principles and applications*, edited by M. Jamshidi. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group. p. 71-105.

Rouse, W.B. 2008. "Health Care as a Complex Adaptive System: Implications for design and management." *The Bridge, National Academy of Engineering*. 38 (1): 17-25.

Sage, A.P. 2000. "Transdisciplinarity Perspectives in Systems Engineering and Management," in *Transdiciplinarity: Recreating Integrated Knowledge*, edited by M.A. Somerville and D. Rappaport. Oxford, UK: EOLSS Publishers. p. 158-169.

von Bertalanffy, L. 1968. *General System Theory: Foundations, Development, Applications*, revised ed. New York, NY, USA: Braziller.

Weinberg, G., and D. Weinberg. 1988. *General Principles of Systems Design*. New York, NY, USA: Dorset House Publishing Company. White, B.E. 2007. "On Interpreting Scale (or View) and Emergence in Complex Systems Engineering." Presented at 1st Annual IEEE Systems Conference, April 9-12, 2007, Honolulu, HI, USA.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

The Enterprise as a System

To enable more efficient and effective enterprise (glossary) transformation, the enterprise needs to be looked at "as a system," rather than as a collection of functions connected solely by information systems and shared facilities (Rouse 2005 and 2009; Lawson 2010). What distinguishes the design (glossary) of enterprise systems from product systems is the inclusion of people as a component (glossary) of the system, not merely as a user/operator of the system.

> *The term 'enterprise system' has taken on a narrow meaning of only the information system an organization uses. Research and project experience has taught us that to design a good enterprise system, we need to adopt a much broader understanding of enterprise systems. The greater view of enterprise systems is inclusive of the processes the system supports, the people who work in the system, and the information [and knowledge] content of the system.* (Giachetti 2010)

It is worth noting that the concept of "service" systems also includes people in the system. The thoughts above do not take this into account, primarily since their perspectives come mainly from a product system experience. The practice of service systems engineering is relatively new and is an emerging discipline. For more information on this, see the articles on Service Systems Engineering.

Creating Value

The primary purpose of an enterprise is to create value for society, other stakeholders, and for the organizations that participate in that enterprise. This is illustrated in Figure 1 that shows all the key elements that contribute to this value creation process. These elements in the enterprise can be treated as a "system" and the processes, methods, and tools ESE can be applied.

There are three types of organizations of interest: businesses, projects, and teams (see note 1). A typical business participates in multiple enterprises through its portfolio of projects. Large SE projects can be enterprises in their own right, with participation by many different businesses, and may be organized as a number of sub-projects.

> *Note 1. The use of the word "business" is not intended to mean only for-profit commercial ventures. As used here, it also includes government agencies and not-for-profit organizations, as well as commercial ventures. Business is the activity of providing goods and services involving financial, commercial, and industrial aspects.*

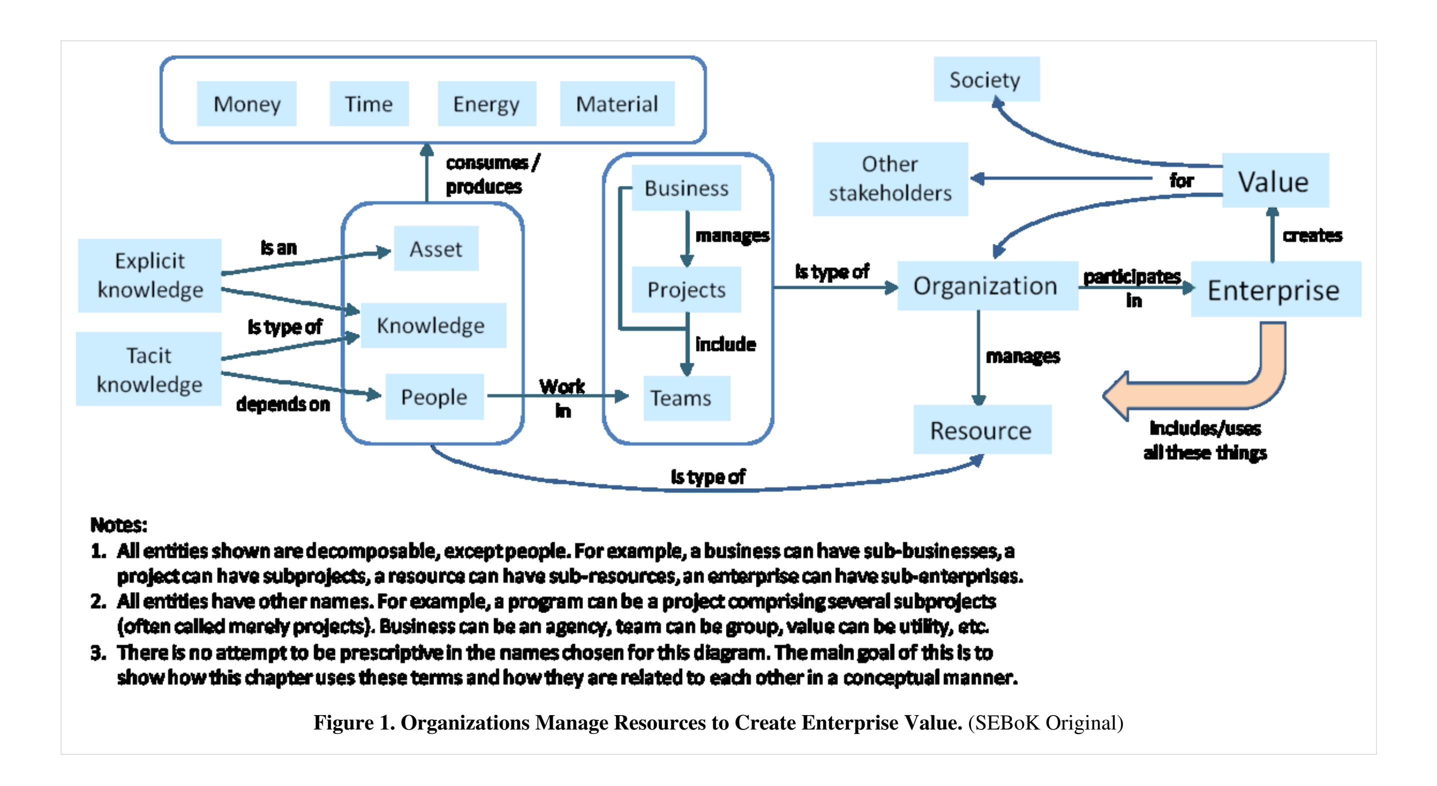


Figure 1. Organizations Manage Resources to Create Enterprise Value. (SEBoK Original)

Resource Optimization

A key choice for businesses that conduct SE is to what extent, if at all, they seek to optimize their use of resources (people, knowledge, assets) across teams, projects, and business units. Optimization of resources is not the goal in itself, but rather a means to achieve the goal of maximizing value for the enterprise and its stakeholders. At one extreme, in a product-oriented organization, projects may be responsible for hiring, training, and firing their own staff, as well as managing all assets required for their delivery of products or services. (The term "product-oriented organization" is not meant in the sense of product-oriented SE, but rather in the sense of this being one of the basic constructs available when formulating organizational strategy.)

At the other extreme, in a functional organization, the projects delegate almost all their work to functional groups. In between these two extremes is a matrix organization that is used to give functional specialists a “home” between project assignments. A full discussion of organizational approaches and situations along with their applicability in enabling SE for the organization is provided in the article called Systems Engineering Organizational Strategy.

The optimization debate can be handled as described in the book called "Enterprise Architecture as Strategy" (Ross, Weill, and Robertson 2006). In other words, an enterprise can choose (or not) to unify its operations and can choose (or not) to unify its information base. There are different strategies the enterprise might adopt to achieve and sustain value creation (and how ESE helps an enterprise to choose). This is further addressed in the section on Enterprise Architecture Formulation & Assessment in the article called Enterprise Capability Management.

Enabling Systems Engineering in the Organization

SE skills, techniques, and resources are relevant to many enterprise functions, and a well-founded SE capability can make a substantial contribution at the enterprise level, as well as at the project level. The article called Systems Engineering Organizational Strategy discusses enabling SE in the organization, while the article called Enabling Businesses and Enterprises focuses on the cross-organizational functions at the business and enterprise levels. The competence of individuals is discussed in the article called Enabling Individuals.

Kinds of Knowledge Used by the Enterprise

Knowledge is a key resource for ESE. There are generally two kinds of knowledge: explicit and tacit. Explicit knowledge can be written down or incorporated in computer codes. Much of the relevant knowledge, however, is “tacit knowledge” that only exists within the heads of people and in the context of relationships that people form with each other (e.g., team, project, and business level knowledge). The ability of an organization to create value is critically dependent on the people it employs, on what they know, how they work together, and how well they are organized and motivated to contribute to the organization’s purpose.

Projects, Programs, and Businesses

The term “program” is used in various ways in different domains. In some domains a team can be called a program (e.g., a customer support team is their customer relationship "program"). In others, an entire business is called a program (e.g., a wireless communications business unit program), and in others the whole enterprise is called a program (e.g., the Joint Strike Fighter program and the Apollo Space program). And in many cases, the terms project and program are used interchangeably with no discernible distinction in their meaning or scope. Typically, but not always, there are program managers who have profit and loss (P&L) responsibility and are the ultimate program decision makers. A program manager may have a portfolio of items (services, products, facilities, intellectual property, etc.) that are usually provided, implemented, or acquired through projects.

The Office of Government Commerce provides a useful distinction between programs and projects:

> *The ultimate goal of a Programme is to realise outcomes and benefits of strategic relevance. To achieve this a programme is designed as a temporary flexible organisation structure created to coordinate, direct and oversee the implementation of a set of related projects and activities in order to deliver outcomes and benefits related to the organisation’s strategic objectives...*
>
> *A programme is likely to have a life that spans several years. A Project is usually of shorter duration (a few months perhaps) and will be focussed on the creation of a set of deliverables within agreed cost, time and quality parameters.* (OGC 2010)

Enabling the Enterprise

ESE, by virtue of its inherent trans-disciplinarity (Sage 2000, 158-169) in dealing with problems that are large in scale and scope, can better enable the enterprise to become more effective and efficient. The complex nature of many enterprise problems and situations usually goes beyond the abilities of standard tools and techniques provided to business school graduates (See also Complexity). ESE can augment the standard business management methods using the tools and methods from the SE discipline to more robustly analyze and evaluate the enterprise as a holistic system. A more general viewpoint, or “view,” for dealing with the enterprise consisting of scale, granularity, mindset, and time frame is provided by White (2007) and by McCarter and White (2009, 71-105).

ESE can provide the enablers to address the concerns of enterprise executives as shown in Table 1 (Rouse 2009). The methods for dealing with, and the special characteristics of, complex adaptive systems must be properly considered when adapting traditional systems engineering (TSE) practices for use at the enterprise level—many of which come out of the systems science and systems thinking domains (von Bertalanffy 1968; Weinberg and Weinberg 1988; Miller and Page 2007; Rouse 2008, 17-25). For an approach to complex adaptive systems (CAS) engineering, refer to White (2009, 1-16) and to McCarter and White (2009, 71-105).

Table 1. Executive Concerns and SE Enablers (Rouse 2009). Reprinted with permission of John Wiley & Sons Inc.

Executive Concerns	SE Enablers
Identifying ends, means, and scope and candidate changes	System complexity analysis to compare "as is" and "to be" enterprises
Evaluating changes in terms of process behaviors and performance	Organizational simulation of process flows and relationships
Assessing economics in terms of investments, operating costs, and returns	Economic modeling in terms of cash flows, volatility, and options
Defining the new enterprise in terms of processes and their integration	Enterprise architecting in terms of workflow, processes, and levels of maturity
Designing a strategy to change the culture for selected changes	Organizational and cultural change via leadership, vision, strategy, and incentives
Developing transformation action plans in terms of what, when, and who	Implementation planning in terms of tasks, schedule, people, and information

Enterprise Engineering

Another distinction is that "enterprise design does not occur at a single point in time like the design of most systems. Instead, enterprises evolve over time and are constantly changing, or are constantly *being designed*" (Giachetti 2010) [emphasis in original]. Giachetti calls this new discipline "enterprise engineering." We consider the enterprise engineering set of practices to be equivalent to what we call enterprise systems engineering (ESE) in this article.

> *The body of knowledge for enterprise engineering is evolving under such titles as enterprise engineering, business engineering, and enterprise architecture . . . . Many systems and software engineering principles are applicable to enterprise engineering, but enterprise engineering's unique complexities require additional principles…. Enterprise engineering's intent is to deliver a targeted level of enterprise performance in terms of shareholder value or customer satisfaction . . . . Enterprise engineering methods include modeling; simulation; total quality management; change management; and bottleneck, cost, workflow, and value-added analysis.* (Joannou 2007)

Supersystem Constructs

System of Systems (SoS)

The phrase "system of systems" (SoS) is commonly used, but there is no widespread agreement on its exact meaning, nor on how it can be distinguished from a conventional system. A system is generally understood to be a collection of elements that interact in such a manner that it exhibits behavior that the elements themselves cannot exhibit. Each element (or component) of the system can be regarded as a system in its own right. Therefore, the phrase "system of systems" can technically be used for any system and, as such, would be a superfluous term. However, the meaning of this phrase has been examined in detail by (Maier 1998, 267-284), and his definition has been adopted by some people (AFSAB 2005). Maier provides this definition:

> *A SoS is an assemblage of components which individually may be regarded as systems, and which possess two additional properties:*

- ***Operational Independence of the Components:*** *If the system-of-systems is disassembled into its component systems the component systems must be able to usefully operate independently. That is, the components fulfill customer-operator purposes on their own; and*
- ***Managerial Independence of the Components:*** *The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a*

continuing operational existence independent of the system-of-systems. (Maier 1998, 267-284)

Maier goes on further saying that "the commonly cited characteristics of systems-of-systems (complexity of the component systems and geographic distribution) are not the appropriate taxonomic classifiers" (Maier 1998, 267-284). Four kinds of SoS have been defined (Dahmann, Lane, and Rebovich 2008).

For further details on SoS, see the *Systems Engineering Guide for SoS* developed by the US Department of Defense (DoD) (DUS(AT) 2008). Also, see the Systems of Systems (SoS) knowledge area.

Federation of Systems (FoS)

Different from the SoS concept, but related to it in several ways, is the concept called "federation of systems" (FoS). This concept might apply when there is a very limited amount of centralized control and authority (Sage and Cuppan 2001, 325-345; Sage and Rouse 2009). Each system in an FoS is very strongly in control of its own destiny, but "chooses" to participate in the FoS for its own good and the good of the "country," so to speak. It is a coalition of the willing. An FoS is generally characterized by significant autonomy, heterogeneity, and geographic distribution or dispersion (Krygiel 1999). Krygiel defined a taxonomy of systems showing the relationships among conventional systems, SoSs, and FOSs.

This taxonomy has three dimensions: autonomy, heterogeneity, and dispersion. A FoS would have a larger value on each of these three dimensions than a non-federated SoS. An "Enterprise System," as described above, could be considered to be an FoS if it rates highly on these three dimensions. However, it is possible for an enterprise to have components that are not highly autonomous, that are relatively homogeneous, and are geographically close together. Therefore, it would be incorrect to say that an enterprise is necessarily the same as an FoS.

Dove points out that in order for a large enterprise to survive in the twenty-first century, it must be more agile and robust (Dove 1999 and 2001). Handy (1992, 59-67) describes a federalist approach called "New Federalism" which identifies the need for structuring of loosely coupled organizations to help them adapt to the rapid changes inherent in the Information Age. This leads to the need for virtual organizations where alliances can be quickly formed to handle the challenges of newly identified threats and a rapidly changing marketplace (Handy 1995, 2-8). Handy sets out to define a number of federalist political principles that could be applicable to an FoS. Handy's principles have been tailored to the domain of systems engineering (SE) and management by Sage and Cuppan (2001, 325-345):

- Subsidiarity,
- Interdependence,
- Uniform and standardized way of doing business,
- Separation of powers,
- Dual citizenship, and
- Scales of SE.

Scales of SE

According to Maier's definition, not every enterprise would be called a SoS since the systems within the enterprise do not usually meet the criteria of operational and managerial independence. In fact, one of the key purposes of an enterprise is to explicitly establish operational dependence between systems that the enterprise owns and/or operates in order to maximize the efficiency and effectiveness of the enterprise as a whole. Therefore, it is more proper to treat an enterprise system and an SoS as different types of things, with different properties and characteristics. This distinction is illustrated in the figure below, where three corresponding categories of SE are shown (DeRosa 2005; Swarz et al. 2006).

It is true that an enterprise can be treated as a system itself and is comprised of many systems within the enterprise, but this discussion will reserve the term SoS to those systems that meet the criteria of operational and managerial independence. This distinction was also used within the MITRE Corporation in their ESE Office (Rebovich and White 2011).

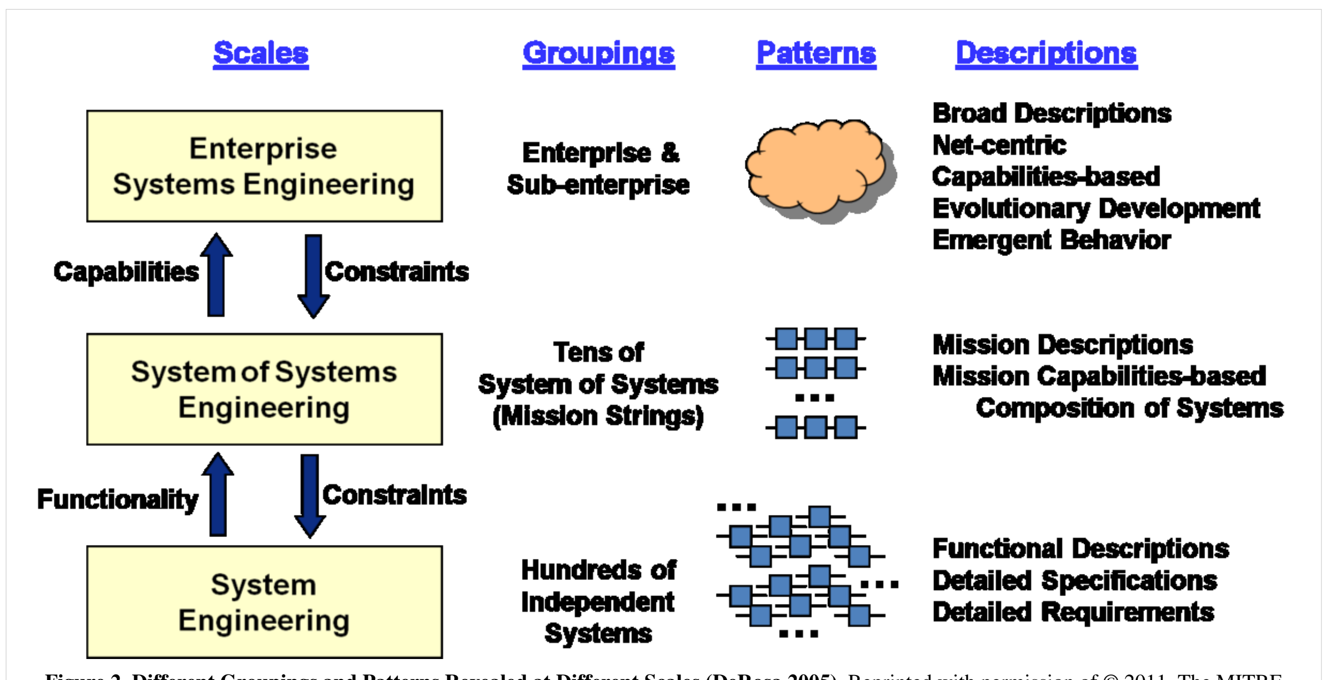


Figure 2. Different Groupings and Patterns Revealed at Different Scales (DeRosa 2005). Reprinted with permission of © 2011. The MITRE Corporation. All Rights Reserved. All other rights are reserved by the copyright owner.

Relationships between Enterprise and SoS

An enterprise may require a particular operational capability that is brought into being by connecting together a chain of systems that together achieve that capability. Any one of these systems in the chain cannot by itself provide this capability. The desired capability is the emergent property of this chain of systems. This chain of systems is sometimes called an SoS. However, the enterprise that requires this capability rarely has direct control over all the systems necessary to provide this full capability. This situation is illustrated in the figure below (Martin 2010).

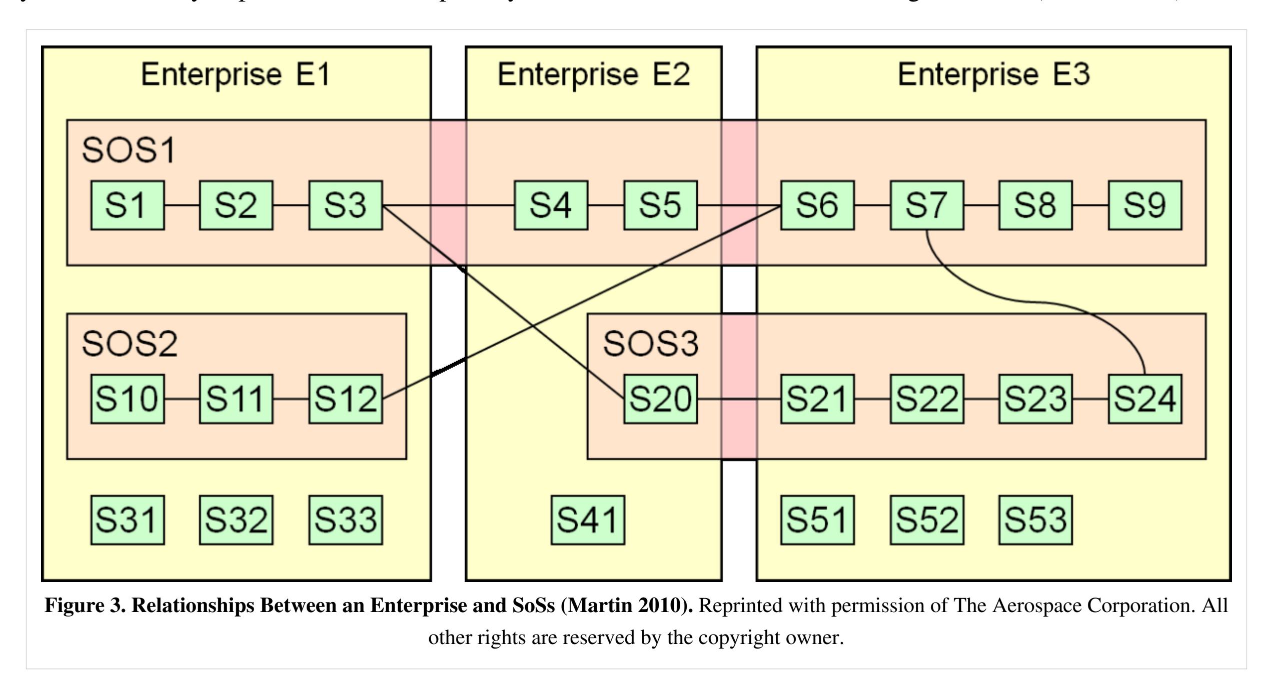


Figure 3. Relationships Between an Enterprise and SoSs (Martin 2010). Reprinted with permission of The Aerospace Corporation. All other rights are reserved by the copyright owner.

Enterprise E1 (in the example above) has full control over SoS2, but not full control over SoS1. TSE can be applied to the individual systems (S1, S2, …, S53) shown within each enterprise, but needs to be augmented with additional activities to handle SoS and enterprise kinds of issues.

There is a general issue regarding dealing with enterprises in this situation: there are at least two enterprises related to any particular SoS. First, there is the enterprise of builders/developers comprising projects and programs, which have to be organized appropriately and adopt special types of architectural principles. Second, there is the enterprise of users (those who use the products and service provided by the first enterprise), which has to exercise its own sort of agility. How the first enterprise designs systems to allow the second to operate is the core issue.

References

Works Cited

AFSAB. 2005. *Report on System-of-Systems Engineering for Air Force Capability Development*. Washington, DC: US Air Force Scientific Advisory Board (AFSAB), US Air Force. SAB-TR-05-04.

Dahmann, J.S., J.A. Lane, and G. Rebovich. 2008. "Systems Engineering for Capabilities." *CROSSTALK: The Journal of Defense Software Engineering*. 21 (11): 4–9.

DeRosa, J.K. 2005. "Enterprise Systems Engineering." Presented at Air Force Association, Industry Day, Day 1, August 4, 2005, Danvers, MA, USA.

Dove, R. 2001. *Response Ability: The Language, Structure, and Culture of the Agile Organization*. New York, NY, USA: John Wiley & Sons.

Dove, R. 1999. "Knowledge Management, Response Ability, and the Agile Enterprise," in Paradigm Shift International [database online]. Accessed September 6, 2011. Available: http:/ / www. parshift. com/ docs/ KmRaAeX.htm.

DUS(AT). 2008. *Systems Engineering Guide for Systems of Systems*, version 1.0. Washington, DC, USA: Deputy Under Secretary of Defense for Acquisition and Technology (DUS(AT)) / US Department of Defense (DoD). Accessed September 6, 2011. Available: http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

Handy, C. 1995. "Trust and the Virtual Organization." *Harvard Business Review*. 73 (3): 2-8.

Handy, C. 1992. "Balancing Corporate Power: A New Federalist Paper." *Harvard Business Review*. 70 (6): 59-67.

Joannou, P. 2007. "Enterprise, Systems, and Software—The Need for Integration." *Computer*. 40 (5): 103-105.

Krygiel, A.J. 1999. *Behind the Wizard's Curtain: An Integration Environment for a System of Systems*. Arlington, VA, USA: C4ISR Cooperative Research Program (CCRP).

Lawson, H. 2010. *A Journey Through the Systems Landscape*. Kings College, UK: College Publications.

Maier, M.W. 1998. "Architecting Principles for Systems-of-Systems." *Systems Engineering, the Journal of the International Council on Systems Engineering (INCOSE). 1 (4): 267-84.*

Martin, J.N. 2010. "An Enterprise Systems Engineering Framework." Presented at 20th Anniversary International Council on Systems Engineering (INCOSE) International Symposium, July 12-15, 2010, Chicago, IL, USA.

McCarter, B.G., and B.E. White. 2009. "Emergence of SoS, sociocognitive aspects," in *Systems of systems engineering: Principles and applications*, edited by M. Jamshidi. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group. p. 71-105.

OGC (Office of Government Commerce). 2010. *Guidelines for Managing Programmes: Understanding programmes and programme management*. London, UK: The Stationery Office.

Ross, J.W., P. Weill, and D. Robertson. 2006. *Enterprise Architecture As Strategy: Creating a Foundation for Business Execution*. Boston, MA, USA: Harvard Business Review Press.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of systems engineering and management,* 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY, USA: Wiley and Sons, Inc.

Rouse, W.B. 2008. "Health Care as a Complex Adaptive System: Implications for design and management." *The Bridge, National Academy of Engineering.* 38 (1): 17-25.

Rouse, W.B. 2005. "Enterprise as Systems: Essential Challenges and Enterprise Transformation." *Systems Engineering*. 8 (2): 138-50.

Sage, A.P. 2000. "Transdisciplinarity Perspectives in Systems Engineering and Management." in *Transdiciplinarity: Recreating Integrated Knowledge*, edited by M.A. Somerville and D. Rappaport. Oxford, UK: EOLSS Publishers. p. 158-169.

Sage, A., and C. Cuppan. 2001. "On the Systems Engineering and Management of Systems of Systems and Federations of Systems." *Information-Knowledge-Systems Management Journal*. 2 (4): 325-345.

Sage, A.P., and W.B. Rouse (eds). 2009. *Handbook of System Engineering and Management*, 2nd ed. New York, NY, USA: John Wiley & Sons.

Swarz, R.S., J.K. DeRosa, and G. Rebovich. 2006. "An Enterprise Systems Engineering Model." Proceedings of the INCOSE International Symposium, July 9-13, 2006, Orlando, FL, USA.

von Bertalanffy, L. 1968. *General System Theory: Foundations, Development, Applications*, revised ed. New York, NY, USA: Braziller.

Weinberg, G., and D. Weinberg. 1988. *General Principles of Systems Design*. New York, NY: Dorset House Publishing Company.

White, B.E. 2007. "On Interpreting Scale (or View) and Emergence in Complex Systems Engineering." Presented at 1st Annual IEEE Systems Conference, 9-12 April, 2007, Honolulu, HI, USA.

Primary References

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

Joannou, P. 2007. "Enterprise, Systems, and Software—The Need for Integration." *Computer*. 40 (5): 103-105.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of systems engineering and management,* 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY, USA: Wiley and Sons, Inc.

Rouse, W.B. 2005. "Enterprise as Systems: Essential Challenges and Enterprise Transformation." *Systems Engineering*. 8 (2): 138-50.

Additional References

Arnold, S., and H. Lawson. 2004. "Viewing Systems From a Business Management Perspective." *Systems Engineering*. 7 (3): 229.

Beimans, F.P.M., M.M. Lankhorst, W.B. Teeuw, and R.G. van de Wetering. 2001. "Dealing with the Complexity of Business Systems Architecting." *Systems Engineering*. 4 (2): 118-33.

Nightingale, D., and D. Rhodes. 2004. "Enterprise systems architecting: Emerging art and science within engineering systems." Presented at Engineering Systems Symposium, Massachusetts Institute of Technology (MIT), 29-31 March, 2004, Boston, MA, USA.

Rebovich, G. 2006. "Systems Thinking for the Enterprise: New & Emerging Perspectives." Presented at IEEE/SMC International Conference on System of Systems Engineering, April 2006, Los Angeles, CA, USA.

Rechtin, E. 1999. *Systems Architecting of Organizations: Why Eagles can't Swim*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

Ring, J. 2004. "Intelligent Enterprises." *INCOSE INSIGHT*. 6 (2).

Ring, J. 2004. "Seeing an Enterprise as a System." *INCOSE INSIGHT*. 6(2).

Valerdi, R., D. Nightingale, and C. Blackburn. 2009. "Enterprises as Systems: Context, Boundaries, and Practical Implications." *Information-Knowledge-Systems Management Journal*. 7 (4): 377-399.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Related Business Activities

The following business (glossary) management activities can be supported by enterprise systems engineering (ESE) activities:

- mission and strategic planning,
- business processes and information Management,
- performance management,
- portfolio management,
- resource allocation and budgeting, and
- program and project management.

Introduction

The figure below shows how these business activities relate to each other as well as the relative scope of ESE and product systems engineering (PSE) (Martin 2010 and 2011). PSE is mainly involved at the project level and collaborates with project management activities, and gets somewhat involved in program management and the resource allocation and budgeting activities. On the other hand, ESE is heavily involved in the higher level activities from the program management level and up. Both ESE and PSE have key roles to play in the enterprise.

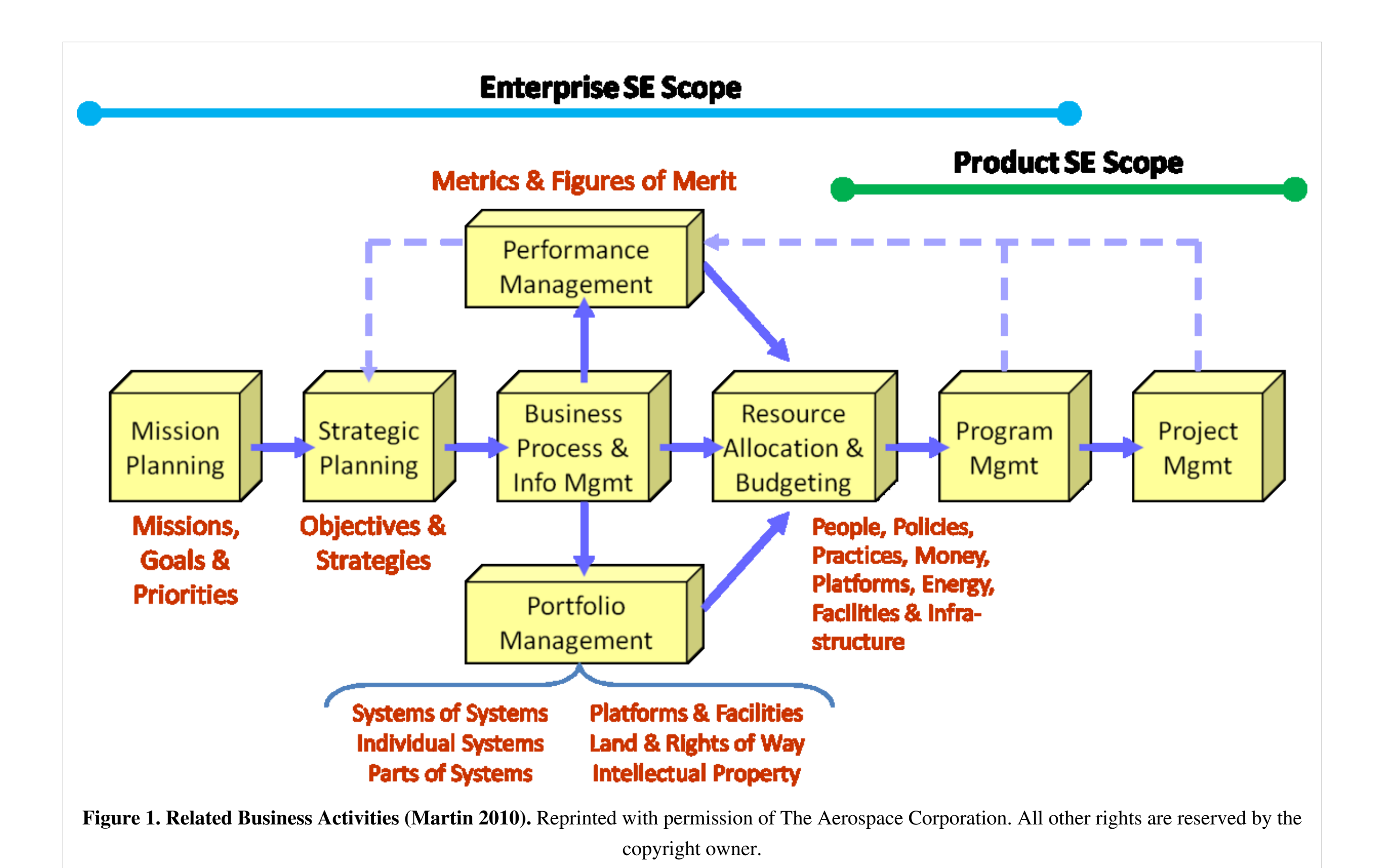


Figure 1. Related Business Activities (Martin 2010). Reprinted with permission of The Aerospace Corporation. All other rights are reserved by the copyright owner.

Shown in this manner, these business activities can be considered to be separate processes with a clear precedence in terms of which process drives other processes. TSE uses “requirements” to specify the essential features and functions of a system. An enterprise, on the other hand, typically uses goals and objectives to specify the fundamental characteristics of desired enterprise operational capabilities. The enterprise objectives and strategies are used in portfolio management to discriminate between options and to select the appropriate balanced portfolio of systems and other enterprise resources.

The first three activities listed above are covered in Enabling Businesses and Enterprises. The other business management activities are described in more detail below in regards to how they relate to ESE.

Business Management Cycles

PDCA stands for "plan-do-check-act" and is a commonly used iterative management process as seen in the figure below. It is also known as the Deming circle or the Shewhart cycle after its two key proponents (Deming 1986; Shewhart 1939). ESE should use the PDCA cycle as one of its fundamental tenets. For example, after ESE develops the enterprise transformation plan, execution of the planned improvements are monitored (i.e., “checked” in the PDCA cycle) to ensure they achieve the targeted performance levels. If not, then action needs to be taken (i.e., “act” in the PDCA cycle) to correct the situation and re-planning may be required. ESE can also use the PDCA cycle in its support of the 'business as usual' efforts, such as the annual budgeting and business development planning activities.

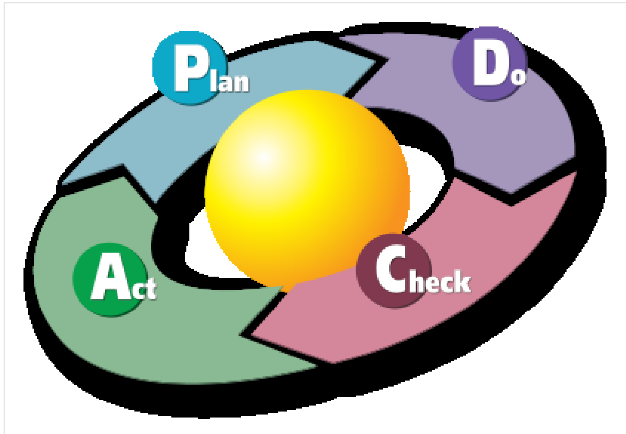


Figure 2. PDCA Cycle. (Source: http://en.wikipedia.org/wiki/PDCA.Accessed July 2010. Diagram by Karn G. Bulsuk (http://blog.bulsuk.com).Used through a CC BY 3.0 license (http://creativecommons.org/licenses/by/3.0/).

It is also worth mentioning the utility of using Boyd's OODA loop (observe, orient, decide, and act) to augment PDCA. This could be accomplished by first using the OODA loop (http://en.wikipedia.org/wiki/OODA_loop), which is continuous in situation awareness, and then followed by using the PDCA approach, which is discrete, having goals, resources, usually time limits, etc. (Lawson 2010).

Portfolio Management

Program and project managers direct their activities as they relate to the systems under their control. Enterprise management, on the other hand, is involved in directing the portfolio of items that are necessary to achieving the enterprise goals and objectives. This can be accomplished by using portfolio management:

> *Project Portfolio Management (PPM) is the centralized management of processes, methods, and technologies used by project managers and project management offices (PMOs) to analyze and collectively manage a group of current or proposed projects based on numerous key characteristics. The objectives of PPM are to determine the optimal resource mix for delivery and to schedule activities to best achieve an organization's operational and financial goals—while honoring constraints imposed by customers, strategic objectives, or external real-world factors.* (http://en.wikipedia.org/wiki/Project_portfolio_management)

The enterprise may not actually own these portfolio items. They could rent or lease these items, or they could have permission to use them through licensing or assignment. The enterprise may only need part of a system (e.g., one bank of switching circuits in a system) or may need an entire system of systems (SoS) (e.g., switching systems, distribution systems, billing systems, provisioning systems, etc.). Notice that the portfolio items are not just those items related to the systems that systems engineering (SE) deals with. These could also include platforms (like ships and oil drilling derricks), facilities (like warehouses and airports), land and rights of way (like railroad property easements and municipal covenants), and intellectual property (like patents and trademarks).

The investment community has been using portfolio management for a long time to manage a set of investments to maximize return for a given level of acceptable risk. These techniques have also been applied to a portfolio of "projects" within the enterprise (Kaplan 2009). However, it should be noted that an enterprise is not merely a portfolio of projects. The enterprise portfolio consists of whatever systems, organizations, facilities, intellectual property, and other resources that are needed to help the enterprise achieve its goals and objectives.

Portfolio management in the context of ESE is well addressed in the following article: http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/enterprise_planning_management/portfolio_management.html (MITRE 2010).

Resource Allocation and Budgeting

The resource allocation and budgeting (RA&B) activity is driven by the portfolio management definition of the optimal set of portfolio elements. Capability gaps are mapped to the elements of the portfolio, and resources are assigned to programs (or other organizational elements) based on the criticality of these gaps. Resources come in the form of people and facilities, policies and practices, money and energy, and platforms and infrastructure. Allocation of resources could also involve the distribution or assignment of corporate assets, like communication bandwidth, manufacturing floor space, computing power, intellectual property licenses, and so on. Resource allocation and budgeting is typically done on an annual basis, but more agile enterprises will make this a more continuous process. Some of the resource allocation decisions deal with base operational organizations that are not project related.

It is sometimes the case that RA&B is part of portfolio management (PfM). But as can be seen in Figure 1, it is sometimes useful and practical to separate these two activities. PfM usually recommends changes to the enterprise portfolio, but RA&B takes these PfM considerations into mind along with inputs from the business process and information management and the performance management activities. Furthermore, PfM is usually an annual or biannual activity whereas RA&B is often done more frequently. RA&B may need to execute *ad hoc* when perturbations happen, such as funding cuts, schedule slips, performance targets missed, strategic goals changed, and so on.

Program and Project Management

Within the enterprise, TSE is typically applied inside a project to engineer a single system (or perhaps a small number of related systems). If there is a SoS or a large, complex individual system to be engineered, then this might be handled at the program level, but is sometimes handled at the project level, depending on the size and complexity of the system-of-interest (See also Complexity).

There are commonly three basic types of projects in an enterprise. A development project takes a conceptual notion of a system and turns this into a realizable design. A production project takes the realizable design for a system and turns this into physical copies (or instantiations). An operations “project” directly operates each system or supports the operation by others. (Base operations are sometimes called "line organizations" and are not typically called projects per se, but should nonetheless be considered as key elements to be considered when adjusting the enterprise portfolio.) The operations project can also be involved in maintaining the system or supporting maintenance by others. A program can have all three types of projects active simultaneously for the same system, as in this example:

- Project A is developing System X version 3.
- Project B is operating and maintaining System X version 2.
- Project C is maintaining System X version 1 in a warehouse as a backup in case of emergencies.

Project management uses TSE as a tool to ensure a well-structured project and to help identify and mitigate cost, schedule, and technical risks involved with system development and implementation. The project level is where the TSE process is most often employed (Martin 1997; ISO/IEC/IEEE 2015; Wasson 2006; INCOSE 2010; Blanchard and Fabrycky 2010).

The Office of Government Commerce provides a useful distinction between programs and projects:

> *The ultimate goal of a Programme is to realise outcomes and benefits of strategic relevance. To achieve this, a programme is designed as a temporary flexible organisation structure created to coordinate, direct and oversee the implementation of a set of related projects and activities in order to deliver outcomes and benefits related to the organisation’s strategic objectives...*
>
> *A programme is likely to have a life that spans several years. A Project is usually of shorter duration (a few months perhaps) and will be focussed on the creation of a set of deliverables within agreed cost, time and quality parameters.* (OGC 2010)

Enterprise Governance

ESE is also concerned with the way in which organizations and embedded management and technical functions work together to achieve success at the enterprise level. Governance frameworks provide the essential additional structure and controls needed to both 'steer a steady ship' (during business as usual) and to 'plot a course to a new place' (during business transformation).

Such frameworks can be designed by recognizing that there are enduring management concerns that need to be addressed and by applying the principle of economy. For example, a particular concern for most organizations is linking the control of projects to business drivers and objectives. This leads to a requirement for a governance body to both approve the initiation of projects, and to regularly review their progress, continuing relevance, and if necessary, mutual coherence in the light of developments inside and outside the enterprise.

This might be achieved by delegating some or all of the roles; depending on circumstances, the enterprise might be driven towards top-down or a more collective, peer-to-peer approach—or even a combination of the two for different functions. Governance bodies and management roles can be engineered in this way against a common set of management concerns. Governance may also include the maintenance of common technical standards and their promulgation and use throughout relevant projects. See Bryant (2012) for more information on governance.

Multi-Level Enterprises

An enterprise does not always have full control over the ESE processes. In some cases, an enterprise may have no direct control over the resources necessary to make programs and projects successful. For example, the Internet Engineering Task Force (IETF) is responsible for the “smooth operation of the Internet,” yet it controls none of the requisite resources.

> *The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. … The actual technical work of the IETF is done in its working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.). Much of the work is handled via mailing lists. The IETF holds meetings three times per year.* (IETF 2010a)

The IETF has “influence” over these resources even though it does not have direct control: "The IETF is unusual in that it exists as a collection of happenings, but is not a corporation and has no board of directors, no members, and no dues" (IETF 2010b).

The ESE processes might be allocated between a “parent” enterprise and “children” enterprises, as shown in the figure below (Martin 2010). The parent enterprise, in this case, has no resources. These resources are owned by the subordinate child enterprises. Therefore, the parent enterprise does not implement the processes of resource allocation and budgeting, program management, and project management.

The parent enterprise may have an explicit contract with the subordinate enterprises, or, as in some cases, there is merely a “working relationship” without the benefit of legal obligations. The parent enterprise will expect performance feedback from the lower level to ensure that it can meet its own objectives. Where the feedback indicates a deviation from the plan, the objectives can be adjusted or the portfolio is modified to compensate.

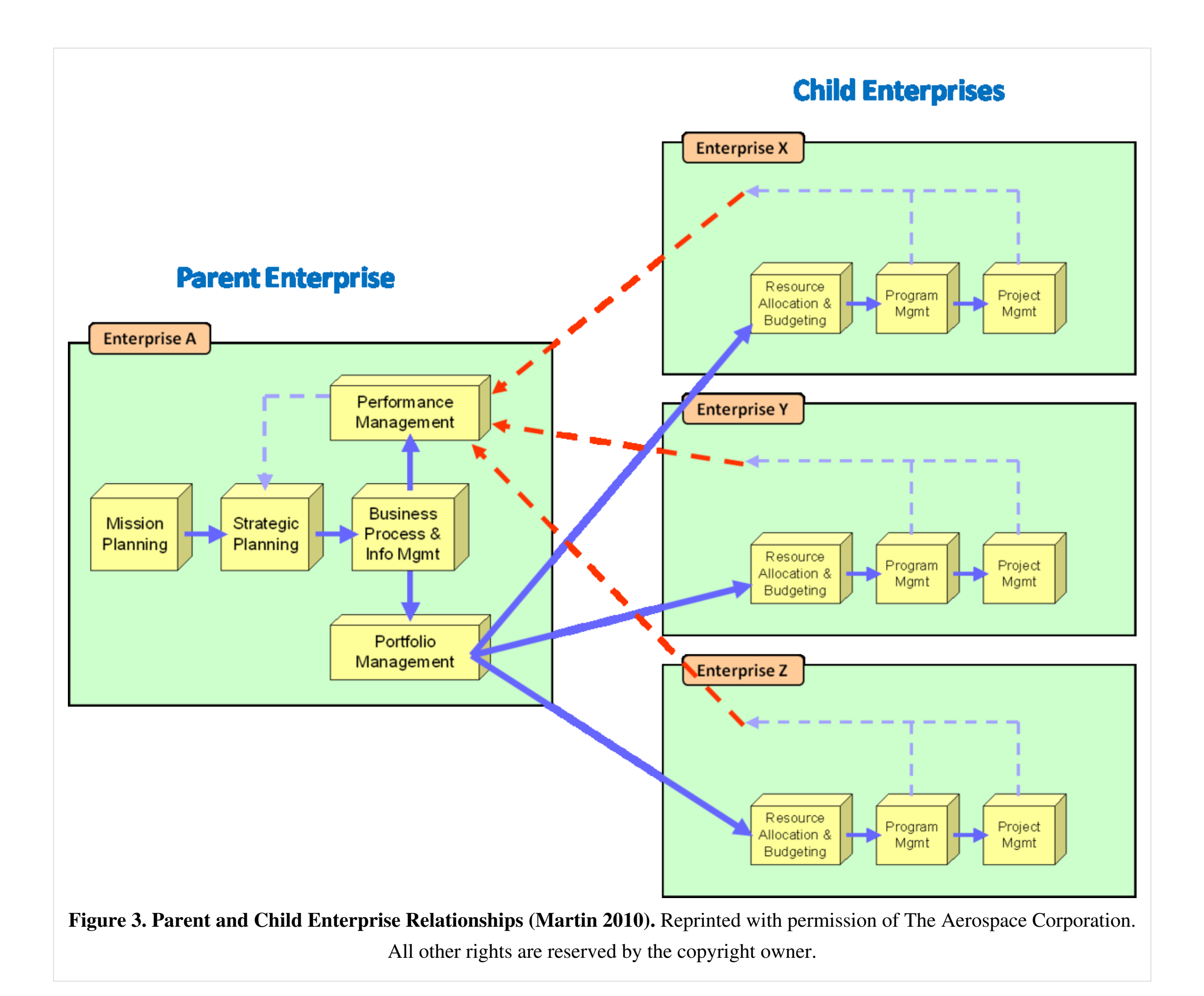


Figure 3. Parent and Child Enterprise Relationships (Martin 2010). Reprinted with permission of The Aerospace Corporation. All other rights are reserved by the copyright owner.

Enterprises X, Y, and Z in the situation shown above will cooperate with each other to the extent that they honor the direction and guidance from the parent enterprise. These enterprises may not even be aware of each other, and, in this case, would be unwittingly cooperating with each other. The situation becomes more complex if each enterprise has its own set of strategic goals and objectives as shown in the figure below.

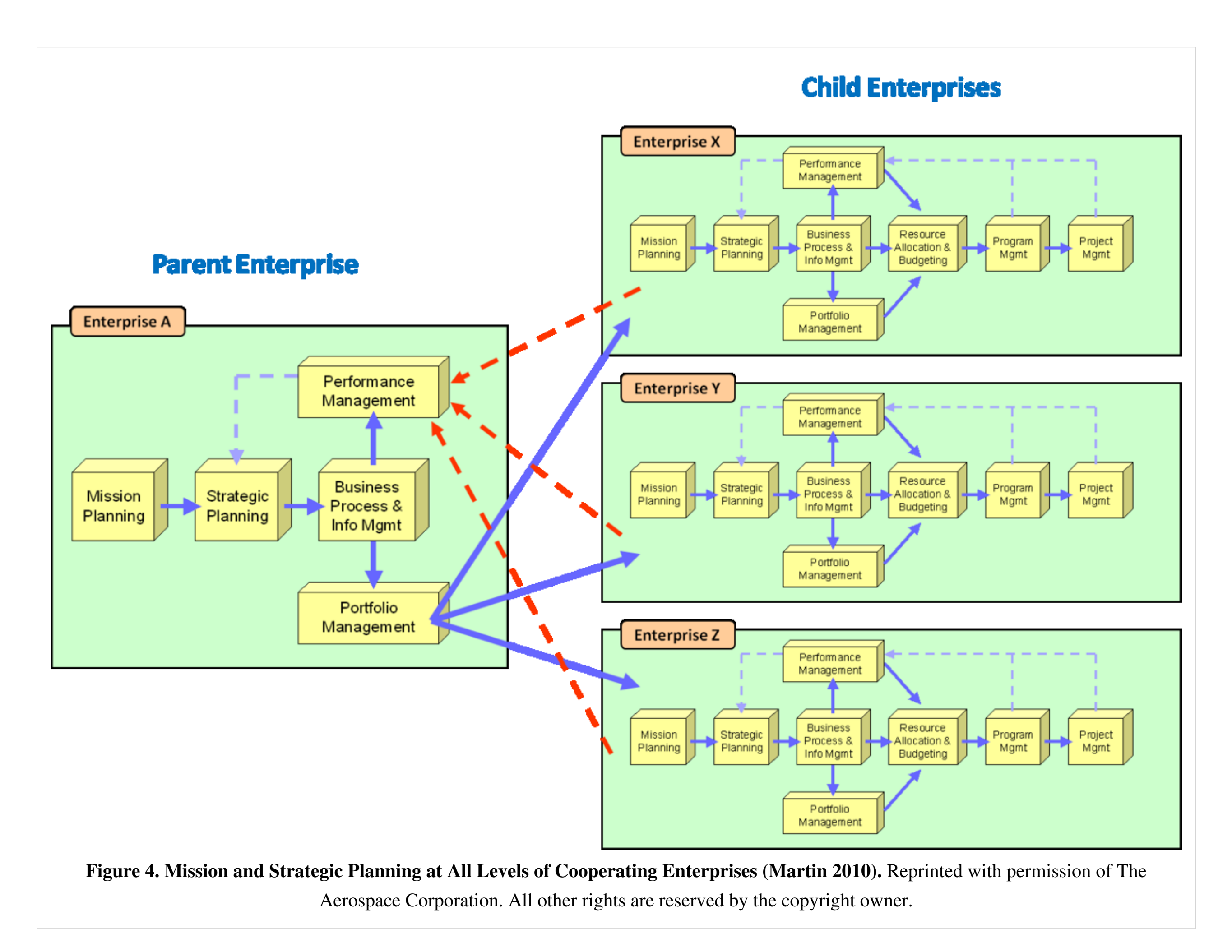


Figure 4. Mission and Strategic Planning at All Levels of Cooperating Enterprises (Martin 2010). Reprinted with permission of The Aerospace Corporation. All other rights are reserved by the copyright owner.

These separate, sub-enterprise objectives will sometimes conflict with the objectives of the parent enterprise. Furthermore, each subordinate enterprise has its own strategic objectives that might conflict with those of its siblings. The situation shown here is not uncommon, and illustrates an enterprise of enterprises, so to speak. This highlights the need for the application of SE at the enterprise level to handle the complex interactions and understand the overall behavior of the enterprise as a whole. TSE practices can be used, to a certain extent, but these need to be expanded to incorporate additional tools and techniques.

References

Works Cited

Bryant, P. 2012. "Modelling Governance within Business Architecture using Topic Mapping." Presented at 22nd Annual International Council on Systems Engineering (INCOSE) International Symposium, July 9-12, 2012, Rome, Italy.

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Englewood Cliffs, NJ, USA: Prentice-Hall.

Deming, W.E. 1986. *Out of the Crisis*. Cambridge, MA, USA: MIT Press, MIT Center for Advance Engineering Study.

IETF. 2010a. "Overview of the IETF," in Internet Engineering Task Force, Internet Society (ISOC) [database online]. Accessed September 6, 2011. Available: http://www.ietf.org/overview.html.

IETF. 2010b. "The Tao of IETF: A Novice's Guide to the Internet Engineering Task Force (draft-hoffman-tao4677bix-10)," in Internet Engineering Task Force, Internet Society (ISOC) [database online].

Accessed September 6, 2011. Available: http://www.ietf.org/tao.html#intro.

INCOSE. 2012. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE 2015. *Systems and Software Engineering -- System Life Cycle Processes.* Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

Kaplan, J. 2009. *Strategic IT portfolio management: Governing enterprise transformation*. Waltham, Massachusetts, USA: Pittiglio, Rabin, Todd & McGrath, Inc. (PRTM).

Lawson, H. 2010. *A Journey Through the Systems Landscape*. Kings College, UK: College Publications.

Martin, J.N. 2011. "Transforming the Enterprise Using a Systems Approach." Presented at 21st Anniversary International Council on Systems Engineering (INCOSE) International Symposium, June 20-23, 2011, Denver, CO, USA.

Martin, J.N. 2010. "An Enterprise Systems Engineering Framework." Presented at 20th Anniversary International Council on Systems Engineering (INCOSE) International Symposium, July 12-15, 2010, Chicago, IL, USA.

Martin, J.N. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products*, 1st ed. Boca Raton, FL, USA: CRC Press.

MITRE. 2012. "Enterprise Engineering," in *Systems Engineering Guide*. Bedford, MA, USA: MITRE Corporation. Accessed July 8, 2012. Available: http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/.

OGC (Office of Government Commerce). 2010. *Guidelines for Managing Programmes: Understanding programmes and programme management*. London, UK: The Stationery Office.

Shewhart, W.A. 1939. *Statistical Method from the Viewpoint of Quality Control*. New York, NY, USA: Dover Publications.

Wasson, C.S. 2006. *System Analysis, Design and Development*. Hoboken, NJ, USA: John Wiley and Sons Ltd.

Primary References

Martin, J.N. 2011. "Transforming the Enterprise Using a Systems Approach." Presented at 21st Anniversary International Council on Systems Engineering (INCOSE) International Symposium, June 20-23, 2011, Denver, CO, USA.

Martin, J.N. 2010. "An Enterprise Systems Engineering Framework." Presented at 20th Anniversary International Council on Systems Engineering (INCOSE) International Symposium, July 12-15, 2010, Chicago, IL, USA.

Additional References

Arnold, S., and H. Lawson. 2004. "Viewing Systems from a Business Management Perspective." *Systems Engineering*. 7 (3): 229.

Beimans, F.P.M., M.M. Lankhorst, W.B. Teeuw, and R.G. van de Wetering. 2001. "Dealing with the Complexity of Business Systems Architecting." *Systems Engineering*. 4 (2): 118-133.

Drucker, P.F. 1994. "The Theory of Business." *Harvard Business Review*. 72 (5): 95-104.

Haeckel, S.H. 2003. "Leading on demand businesses–Executives as architects." *IBM Systems Journal*. 42 (3): 405-13.

Kaplan, R., and D. Norton. 1996. *The balanced scorecard: Translating strategy into action*. Cambridge, MA, USA: Harvard Business School Press.

Lissack, M.R. 2000. "Complexity Metaphors and the Management of a Knowledge Based Enterprise: An Exploration of Discovery." PhD Dissertation in Business Administration. Henley-on-Thames, UK: Henley Management College, University of Reading.

Rechtin, E. 1999. *Systems Architecting of Organizations: Why Eagles Can't Swim*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Enterprise Systems Engineering Key Concepts

The purpose of traditional systems engineering (TSE) is to bring together a diversity of discipline experts to address a wide range of problems inherent in the development of a large, complex (glossary) “single” system (glossary) (Blanchard and Fabrycky 2010; Hall 1989; Sage and Rouse 2009). Enterprise systems engineering (ESE) expands beyond this traditional basis to “consider the full range of SE services increasingly needed in a modern organization where information-intensive systems are becoming central elements of the organization’s business strategy” (Carlock and Fenton 2001, 242-261). The traditional role of systems engineering (SE) is heavily involved in system acquisition and implementation, especially in the context of government acquisition of very large, complex military and civil systems (e.g., F22 fighter jet and air traffic control systems).

ESE encompasses this traditional role in system acquisition, but also incorporates enterprise strategic planning and enterprise investment analysis (along with others as described below). These two additional roles for SE at the enterprise level are “shared with the organization’s senior line management, and tend to be more entrepreneurial, business-driven, and economic in nature in comparison to the more technical nature of classical systems engineering” (Carlock and Fenton 2001, 242-261).

Closing the Gap

ESE practices have undergone significant development recently.

> *Today the watchword is enterprise systems engineering, reflecting a growing recognition that an 'enterprise' may comprise many organizations from different parts of government, from the private and public sectors, and, in some cases, from other nations.* (MITRE 2004)

Rebovich (2006) says there are “new and emerging modes of thought that are increasingly being recognized as essential to successful systems engineering in enterprises.” For example, in addition to the TSE process areas, MITRE has included the following process areas in their ESE process (DeRosa 2005) to close the gap between ESE and PSE:

- strategic technical planning,
- enterprise architecture,
- capabilities-based planning analysis,
- technology planning, and
- enterprise analysis and assessment.

These ESE processes are shown in the context of the entire enterprise in the figure below (DeRosa 2006). The ESE processes are shown in the middle with business processes on the left and TSE processes on the right. These business processes are described in the article called Related Business Activities. The TSE processes are well documented in many sources, especially in the ISO/IEC/IEEE 15288 standard (2015).

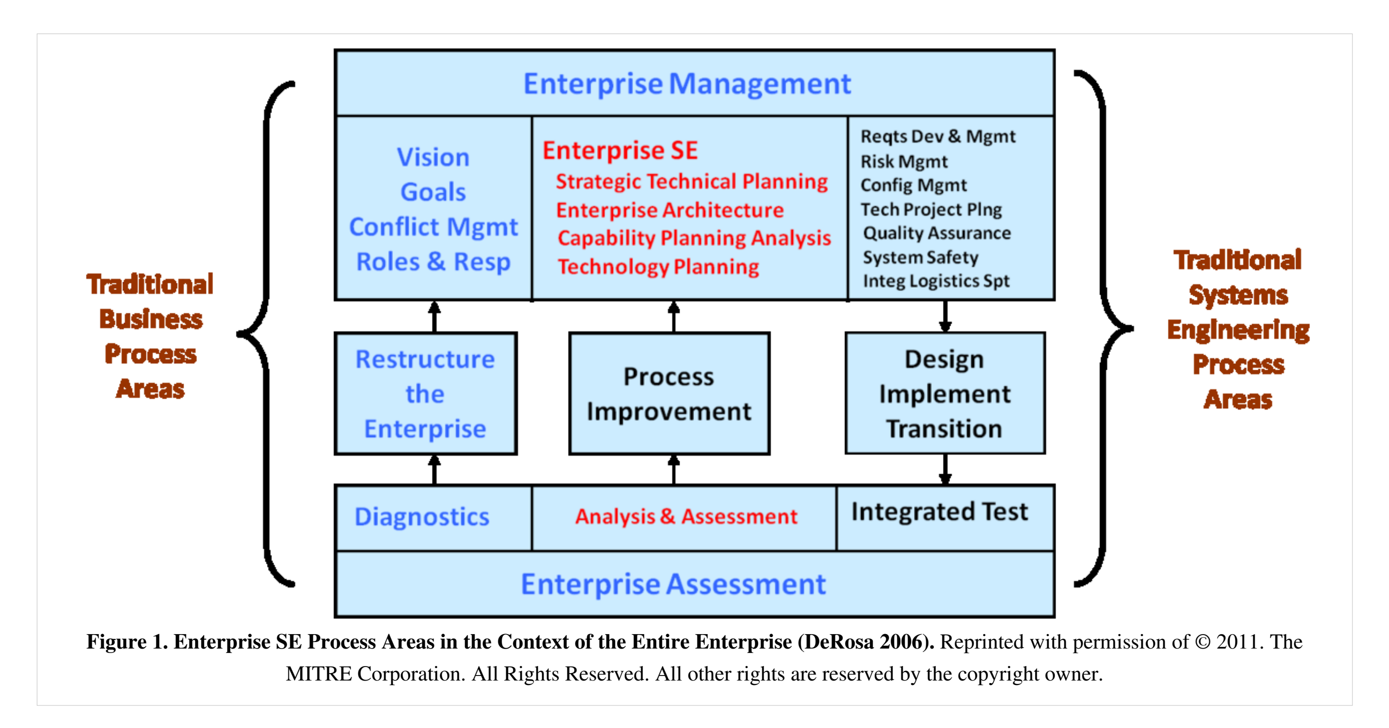


Figure 1. Enterprise SE Process Areas in the Context of the Entire Enterprise (DeRosa 2006). Reprinted with permission of © 2011. The MITRE Corporation. All Rights Reserved. All other rights are reserved by the copyright owner.

SE is viewed by many organizations and depicted in many process definitions as bounded by the beginning and end of a system development project. In MITRE, this restricted definition was referred to as TSE. Many have taken a wider view seeking to apply SE to the "whole system" and "whole life cycle." For example, Hitchins (1993) sets out a holistic, whole-life, wider system view of SE centered on operational purpose. Elliott and Deasley (2007) discuss the differences between development phase SE and in-service SE.

In contrast to TSE, the ESE discipline is more like a "regimen" (Kuras and White 2005) that is responsible for identifying "outcome spaces," shaping the development environment, coupling development to operations, and rewarding results rather than perceived promises (DeRosa 2005). ESE must continually characterize the operational environmental and the results of enterprise or SoS interventions to stimulate further actions within and among various systems in the enterprise portfolio. Outcome spaces are characterized by a set of desired capabilities that help meet enterprise objectives, as opposed to definitive "user requirements" based on near-term needs. Enterprise capabilities must be robust enough to handle unknown threats and situations in the future. A detailed description of previous MITRE views on ESE can be found in a work by Rebovich and White (2011).

Role of Requirements in ESE

TSE typically translates user needs into system requirements that drive the design of the system elements. The system requirements must be "frozen" long enough for the system components to be designed, developed, tested, built, and delivered to the end users (which can sometimes take years, and in the case of very large, complicated systems like spacecraft and fighter jets, more than a decade).

ESE, on the other hand, must account for the fact that the enterprise must be driven not by requirements (that rarely can even be defined, let alone made stable), but instead by continually changing organizational visions, goals, governance priorities, evolving technologies, and user expectations. An enterprise consists of people, processes, and technology where the people act as "agents" of the enterprise:

> *Ackoff has characterized an enterprise as a 'purposeful system' composed of agents who choose both their goals and the means for accomplishing those goals. The variety of people, organizations, and their strategies is what creates the inherent complexity and non-determinism in an enterprise. ESE must account for the concerns, interests and objectives of these agents.* (Swarz, DeRosa, and Rebovich 2006)
> (See also Complexity)

Whereas TSE focuses on output-based methodologies (e.g., functional analysis and object-oriented analysis), ESE is obligated to emphasize outcomes (e.g., business analysis and mission needs analysis), especially those related to the enterprise goals and key mission needs.

Enterprise Entities and Relationships

An enterprise “system” has different entities and relationships than you might find in a product/service system (see note 1). These can be usefully grouped into two categories: asset items and conceptual items. An example of an asset is hardware and software. Examples of conceptual items are things like analysis, financial elements, markets, policies, process, and strategy.

Note 1. An “enterprise system” should not be confused with the enterprise “perceived as a system.” An enterprise system is a product (or service) system used across the enterprise, such as payroll, financial accounting, or enterprise resource planning applications, and consolidated data center, data warehouse, and other such facilities and equipment used across one or more organizations.

Products and services are sometimes treated as “assets” as shown in the figure below (Troux 2010). This categorization of enterprise items comes from the semantic model (i.e., metamodel) used in the Troux Architect modeling tool for characterization and analysis of an enterprise architecture. Other enterprise entities of interest are things like information, knowledge, skills, finances, policies, process, strategy, markets, and resources, but these are categorized as "concept" items (in this particular schema). Further details on how to use this metamodel's entities and relationships are provided by Reese (2010).

Table 1. Asset Domain and Concept Domain Categories for Enterprise Entities. (Troux 2010) Reprinted with permission of Copyright © 2010 Troux Technologies. All other rights are reserved by the copyright owner.

Asset Domains	Concept Domains
Application and Software Domain Data Domain Document Domain Infrastructure and Hardware Domain IT Product Domain IT Service Domain Location Domain Organization Domain Product and Service Domain Services Portfolio Management Domain	Analysis Domain Financial Domain General Domain Information Domain IT Architecture Domain Knowledge and Skill Domain Market Domain Policy Domain Process Domain Resource Domain Strategy Domain Timeline Domain Transition Domain

The application/software and infrastructure/hardware domains are likely the most familiar to systems engineers (as illustrated in the figure below). The application/software domain contains things like the deployed software itself, plus applications, modules, servers, patches, functions, and messages. The infrastructure/hardware domain contains things like the hardware itself, plus networks and different kinds of hardware like computing hardware, cabinets, and network devices. There might be different subtypes of computing hardware like computers, servers, desktops, laptops, and mainframes. You can see from this elaboration of these domains that an enterprise architecture "schema" can be quite extensive in the kinds of things it can model.

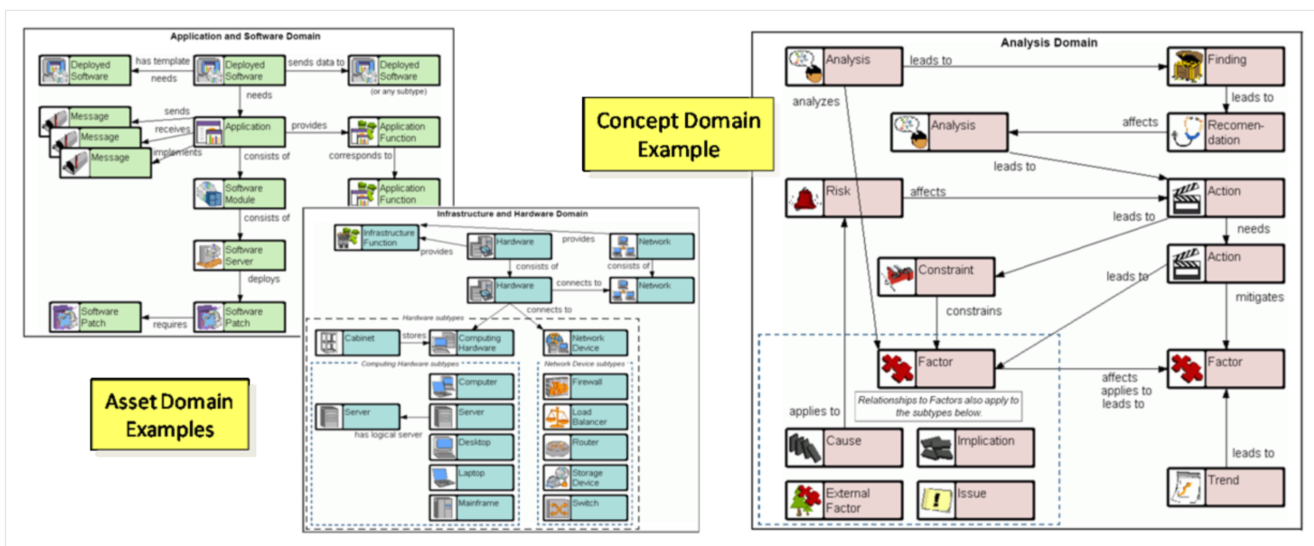


Figure 2. Example of Enterprise Entities & Relationships (Troux 2010). Reprinted with permission of Copyright © 2010 Troux Technologies. All other rights are reserved by the copyright owner.

The less technical domains would be things like policy, market, strategy, transition, financial, knowledge and skill, and analysis. In a typical enterprise architecture schema like this, there could be over a hundred types of modeling objects grouped into these domains. The examples give above are from the Troux Semantics metamodel used in the Troux Architect modeling tool for enterprise architecture activities. Other enterprise modeling tools have similar metamodels (sometimes called "schemas"). See Reese (2010) for more details on how to use the metamodel shown in the figure above.

Enterprise Architecture Frameworks & Methodologies

Enterprise architecture frameworks are collections of standardized viewpoints, views, and models that can be used when developing architectural descriptions of the enterprise. These architecture descriptions can be informal, based on simple graphics and tables, or formal, based on more rigorous modeling tools and methods. ISO/IEC 42010 (2011) specifies how to create architecture descriptions.

These frameworks relate to descriptive models of an enterprise, with conventions agreed in particular communities. There are various frameworks and methodologies available that assist in the development of an enterprise architecture.

Urbaczewski and Mrdalj (2006) provide an overview and comparison of five prominent architectural frameworks, including:

- the Zachman Framework for Enterprise Architecture (Zachman 1992),
- the Department of Defense Architecture Framework (DoDAF) (DoD 2010),
- the Federal Enterprise Architecture Framework (FEAF) (FEA 2001),
- the Treasury Enterprise Architecture Framework (TEAF) (US Treasury 2000),
- and The Open Group Architectural Framework (TOGAF) (TOGAF 2009).

References

Works Cited

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Englewood Cliffs, NJ, USA: Prentice-Hall.

Carlock, P., and R. Fenton. 2001. "System of Systems (SoS) Enterprise Systems Engineering for Information-Intensive Organizations." *Systems Engineering*. 4 (4): 242-261.

CIO Council. 1999. *Federal Enterprise Architecture Framework (FEAF)*, version 1.1. Washington, DC, USA: Federal Chief Information Officers Council.

DeRosa, J.K. 2005. "Enterprise Systems Engineering." Presented at Air Force Association, Industry Day, Day 1, August 4, 2005, Danvers, MA, USA.

DoD. 2010. *DoD Architecture Framework (DoDAF)*, version 2.0. Washington, DC: U.S. Department of Defense (DoD).

Elliott, C., and P. Deasley. 2007. *Creating Systems that Work--Principles of Engineering Systems for the 21st Century*. London, England, UK: Royal Academy of Engineering.

FEA. 2001. "Federal Enterprise Architecture – Practical Guide, version 1.0, February 2001." Available: https:// secure.cio.noaa.gov/hpcc/docita/files/a_practical_guide_to_federal_enterprise_architecture.pdf.

Friedman, G., and A.P. Sage. 2004. "Case Studies of Systems Engineering and Management in Systems Acquisition." *Systems Engineering*. 7 (1): 84-96.

Hall, A.D. 1989. *Metasystems Methodology: A New Synthesis and Unification*, 1st ed. Oxford, UK: Pergamon Press.

Hitchins, D. 1993. *Putting Systems to Work*. New York, NY, USA: John Wiley & Sons.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Kuras, M.L., and B.E. White. 2005. "Engineering Enterprises Using Complex-Systems Engineering." Annotated presentation at 15th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY, USA.

MITRE. 2004. *MITRE 2004 Annual Report". McLean, VA, USA: MITRE Corporation.*

Rebovich, G. 2006. "Systems Thinking for the Enterprise: New & Emerging Perspectives." Presented at IEEE/SMC International Conference on System of Systems Engineering, April 2006, Los Angeles, CA, USA.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

Reese, R.J. 2010. *Troux Enterprise Architecture Solutions*. Birmingham, UK: Packt Publishing Ltd.

Sage, A.P., and W.B. Rouse (eds). 2009. *Handbook of System Engineering and Management*, 2nd ed. New York, NY, USA: John Wiley & Sons.

Swarz, R.S., J.K. DeRosa, and G. Rebovich. 2006. "An Enterprise Systems Engineering Model." Proceedings of the 16th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2006, Orlando, FL, USA.

TOGAF. 2009. "The Open Group Architecture Framework," version 9. Accessed September 7, 2011. Available: http://www.opengroup.org/togaf/.

Troux. 2010. *Metamodeling and modeling with Troux Semantics*, version 9. Austin, TX, USA: Troux Technologies.

Urbaczewski, L., and S. Mrdalj. 2006. "A Comparison of Enterprise Architecture Frameworks." *Issues in Information Systems*. 7 (2): 18-26.

US Treasury. 2000. *Treasury Enterprise Architecture Framework,* version 1. Washington, DC, USA: US Department of the Treasury Chief Information Officer Council.

Zachman, J.A. 1992. "Extending and Formalizing the Framework for Information Systems Architecture." *IBM Systems Journal*. 31 (3): 590-616.

Zachman, J.A. 1987. "A Framework for Information Systems Architectures." *IBM Systems Journal*. 26 (3): 276-92.

Primary References

Kuras, M.L., and B.E. White. 2005. "Engineering Enterprises Using Complex-Systems Engineering." Annotated presentation at 15th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY, USA.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

Swarz, R.S., J.K. DeRosa, and G. Rebovich. 2006. "An Enterprise Systems Engineering Model." Proceedings of the 16th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2006, Orlando, FL, USA.

Additional References

Journal of Enterprise Architecture. Available: http://www.globalaea.org/?page=JEAOverview.

Minoli, D. 2008. *Enterprise Architecture A to Z: Frameworks, Business Process Modeling, SOA, and Infrastructure Technology*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group, An Auerbach Book.

TRAK. 2011. "TRAK Enterprise Architecture Framework." Accessed September 7, 2011. Available: http://trak.sourceforge.net/index.html.

Vernadat, F.B. 1996. *Enterprise Modelling and Integration - Principles and Applications*. London, UK: Chapman and Hall.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Enterprise Systems Engineering Process Activities

The application of the key concepts of Enterprise Systems Engineering requires processes. These processes span and can transform the enterprise.

Systems Engineering Role in Transforming the Enterprise

Enabling Systematic Enterprise Change

The systems engineering (SE) process as applied to the enterprise as a whole could be used as the "means for producing change in the enterprise … [where the] … Seven Levels of change in an organization [are defined] as effectiveness, efficiency, improving, cutting, copying, differentiating and achieving the impossible" (McCaughin and DeRosa 2006). The essential nature of enterprise systems engineering (ESE) is that it "determines the balance between complexity and order and in turn the balance between effectiveness and efficiency. When viewed as the fundamental mechanism for change, it goes beyond efficiency and drives adaptation of the enterprise" (McCaughin and DeRosa 2006). McCaughin and DeRosa (2006) provide a reasonably good definition for an enterprise that captures this notion of balance:

> *Enterprise: People, processes and technology interacting with other people, processes and technology, serving some combination of their own objectives, those of their individual organizations and those of the enterprise as a whole.*

Balancing Effectiveness versus Efficiency

Ackoff tells us that:

> *Data, information, knowledge and understanding enable us to increase efficiency, not effectiveness. The value of the objective pursued is not relevant in determining efficiency, but it is relevant in determining effectiveness. Effectiveness is evaluated efficiency. It is efficiency multiplied by value.* ***Intelligence*** *is the* ***ability to increase efficiency****;* ***wisdom*** *is the* ***ability to increase effectiveness****.*
>
> *The difference between efficiency and effectiveness is reflected in the difference between development and growth. Growth does not require an increase in value; development does. Therefore, development requires an increase in* ***wisdom as well as understanding, knowledge and information****.* ((Ackoff 1989, 3-9), emphasis added)

ESE has a key role to play in establishing the right balance between effectiveness and efficiency in enterprise operations and management. Value stream analysis is one technique, among others, that can help ESE determine where inefficiencies exist or ineffective results are being achieved.

Value Stream Analysis

Value stream analysis is one way of treating the enterprise as a system. It provides insights regarding where in the sequence of enterprise activities value is added as it moves towards the final delivery to customer or user (Rother and Shook 1999). It relates each step to the costs entailed in that step in terms of resource consumption (i.e., money, time, energy, and materials). In addition to direct costs, there may also be indirect costs due to overhead factors or infrastructure elements. This activity commonly involves drawing a flowchart of the value stream for the enterprise as illustrated in the figure below.

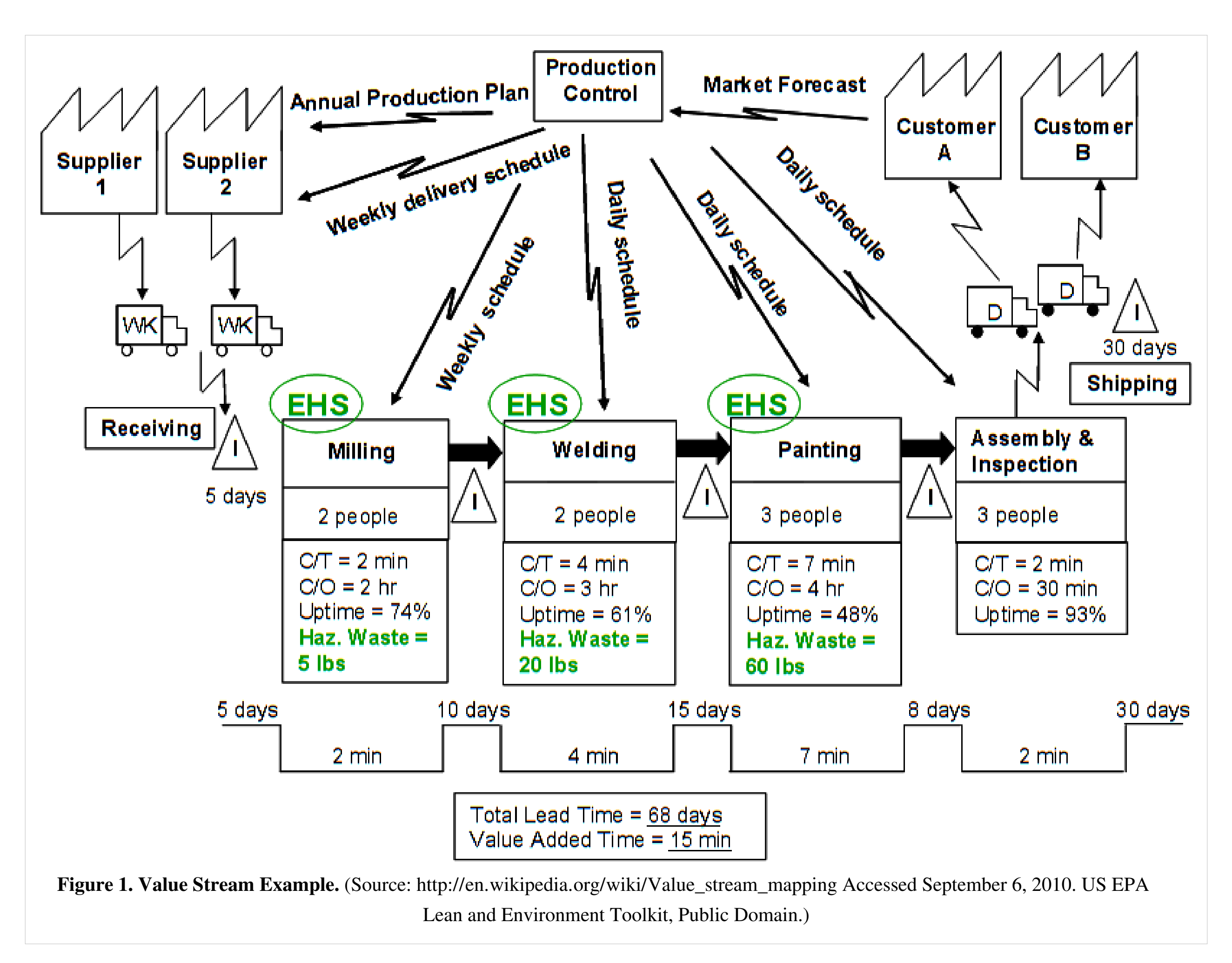


Figure 1. Value Stream Example. (Source: http://en.wikipedia.org/wiki/Value_stream_mapping Accessed September 6, 2010. US EPA Lean and Environment Toolkit, Public Domain.)

Analysis of this value stream diagram can highlight unnecessary space, excessive distance traveled, processing inefficiencies, and so on. Value stream mapping is associated with so-called “lean enterprise” initiatives. At Toyota, where the technique originated, it is known as “material and information mapping” (Rother 2009). Various value stream mapping tools are available (Hines and Rich 1997).

Enterprise Management Process Areas

Martin (2010) has determined that the following four processes are needed in ESE beyond the traditional SE processes in support of enterprise management activities:

1. Strategic technical planning,
2. Capability-based planning analysis,
3. Technology and standards planning, and
4. Enterprise evaluation and assessment.

The interactions between these four processes are illustrated below, along with their interactions with other processes that deal with architecture, requirements, risk, and opportunity.

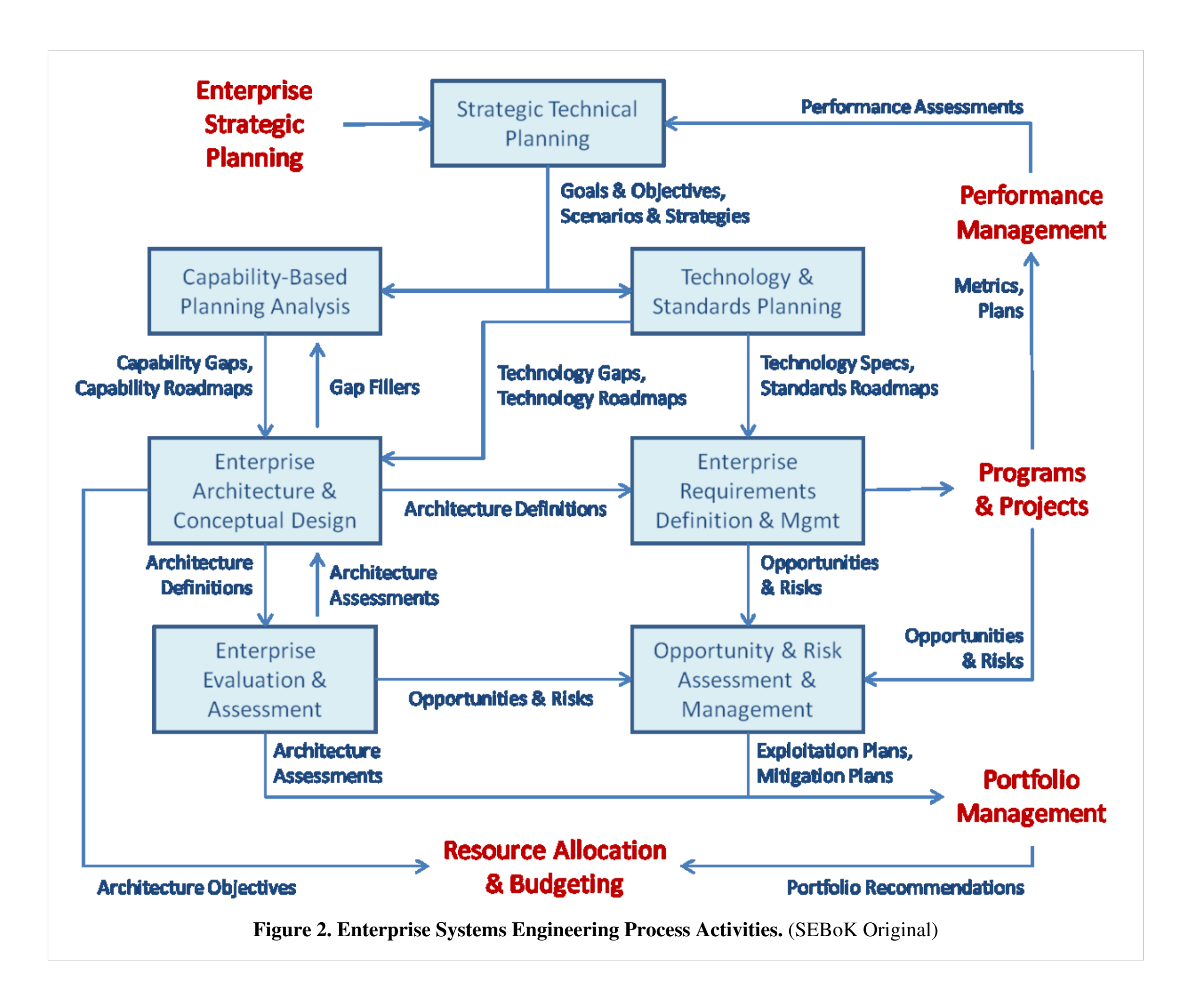


Figure 2. Enterprise Systems Engineering Process Activities. (SEBoK Original)

Strategic Technical Planning

The purpose of strategic technical planning (STP) is to establish the overall technical strategy for the enterprise. It creates the balance between the adoption of standards (see also Systems Engineering Standards) and the use of new technologies, along with consideration of the people aspects driven by the relevant trans-disciplinary technical principles and practices from psychology, sociology, organizational change management, etc.

This process uses the roadmaps developed during technology and standards planning (TSP). It then maps these technologies and standards against the capabilities roadmap to determine potential alignment and synergy. Furthermore, lack of alignment and synergy is identified as a risk to avoid or an opportunity to pursue in the technical strategy. The technical strategy is defined in terms of implementation guidance for the programs and projects.

One reason that STP and TSP are separate processes is that they are often done by different groups in the enterprise and they involve different skill sets. TSP is often done by the technology and science groups. TSP is done closer to (if not in) the chief architect and budget planning groups. Sometimes the great technology proposed by TSP just doesn't line up with the capabilities needed in the requisite time frame. STP does this balancing between technology push and capability pull.

Capability-Based Planning Analysis

The purpose of *Capability-based Planning Analysis* is to translate the enterprise vision and goals into a set of current and future capabilities that helps achieve those goals. Current missions are analyzed to determine their suitability in supporting the enterprise goals. Potential future missions are examined to determine how they can help achieve the vision. Current and projected capabilities are assessed to identify capability gaps that prevent the vision and technical strategy from being achieved. These capability gaps are then used to assess program, project, and system opportunities that should be pursued by the enterprise. This is defined in terms of success criteria of what the enterprise is desired to achieve.

There are different types of capabilities, as shown in the figure below. It is common practice to describe capabilities in the form of capability hierarchies and capability roadmaps. Technology roadmaps (discussed below under Technology Planning) are usually related to the system capabilities while business capability roadmaps (BCRMs) are related to the operational capabilities of the enterprise as a whole (ref: Business-Capability Mapping: Staying Ahead of the Joneses, http://msdn.microsoft.com/en-us/library/bb402954.aspx). The BCRM development is usually done as part of enterprise strategic planning, which is one level higher than, and a key driver for, the strategic technical planning activity described above.

In some domains there may be competency roadmaps dealing with the organizational capabilities, with perhaps the desired competency levels of individuals mapped out in terms of the jobs or roles used in the enterprise or perhaps in terms of the knowledge and skills required for certain activities. (For more information on systems engineering competency, see the Enabling Individuals article.)

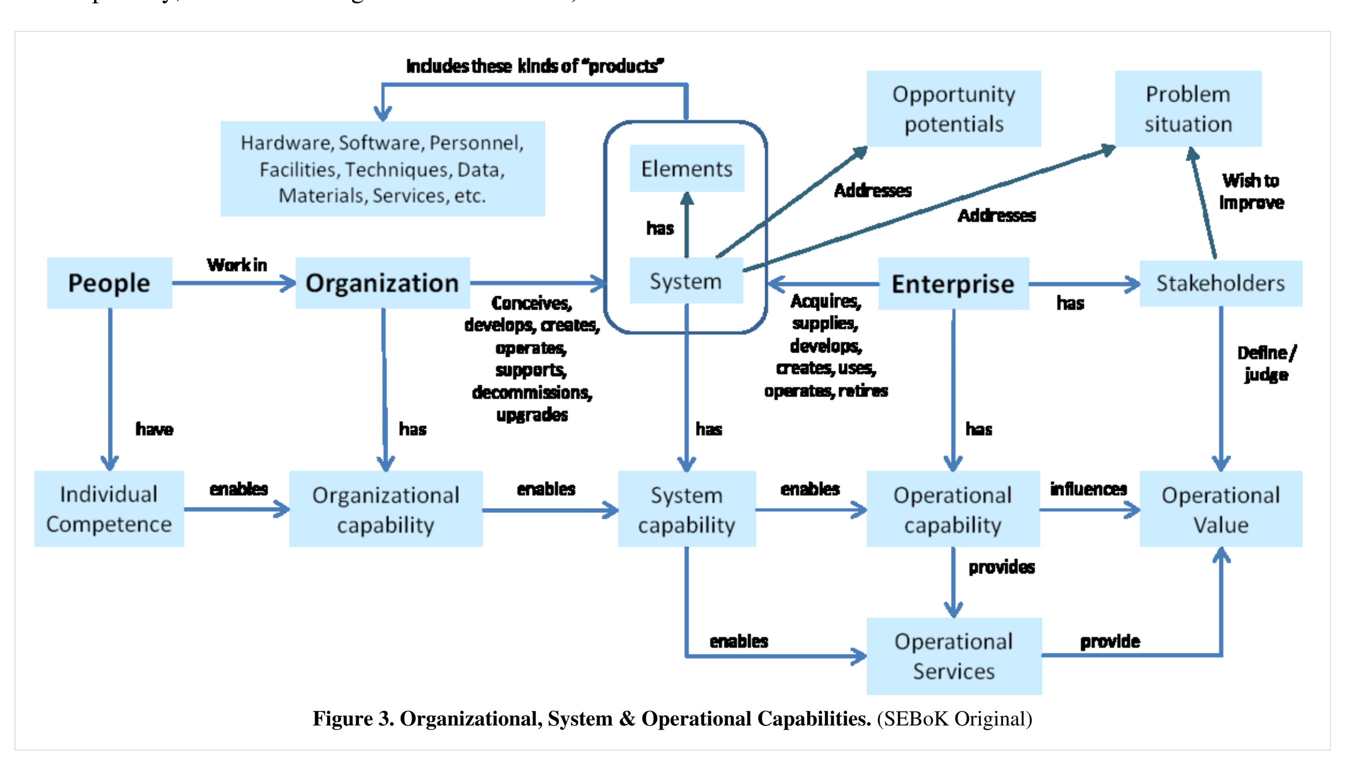


Figure 3. Organizational, System & Operational Capabilities. (SEBoK Original)

Technology and Standards Planning

The purpose of *Technology Planning* is to characterize technology trends in the commercial marketplace and the research community. This activity covers not just trend identification and analysis, but also technology development and transition of technology into programs and projects. It identifies current, and predicts future, technology readiness levels for the key technologies of interest. Using this information, it defines technology roadmaps. This activity helps establish the technical strategy and implementation guidance in the strategic technical plan. The business capabilities roadmap (BCRM) from the strategic planning activity is used to identify which technologies can contribute to achieved targeted levels of performance improvements.

The purpose of *Standards Planning* is to assess technical standards to determine how they inhibit or enhance the incorporation of new technologies into systems development projects. The future of key standards is forecast to determine where they are headed and the alignment of these new standards with the life cycles for the systems in the enterprise's current and projected future portfolios. The needs for new or updated standards are defined and resources are identified that can address these needs. Standardization activities that can support development of new or updated standards are identified (See also Systems Engineering Standards).

Enterprise Evaluation and Assessment

The purpose of enterprise evaluation and assessment (EE&A) is to determine if the enterprise is heading in the right direction. It does this by measuring progress towards realizing the enterprise vision. This process helps to "shape the environment" and to select among the program, project, and system opportunities. This is the primary means by which the technical dimensions of the enterprise are integrated into the business decisions.

This process establishes a measurement program as the means for collecting data for use in the evaluation and assessment of the enterprise. These measures help determine whether the strategy and its implementation are working as intended. Measures are projected into the future as the basis for determining discrepancies between what is observed and what had been predicted to occur. This process helps to identify risks and opportunities, diagnose problems, and prescribe appropriate actions. Sensitivity analysis is performed to determine the degree of robustness and agility of the enterprise.

Roberts states that EE&A must go beyond traditional system evaluation and assessment practices (Roberts 2006). He says that this process area:

> *must de-emphasize the utility of comparing detailed metrics against specific individual requirement values, whether the metrics are derived from measurement, simulation or estimation… [it] must instead look for break points where capabilities are either significantly enhanced or totally disabled.*

Key characteristics of this activity are the following:

- Multi-scale analysis,
- Early and continuous operational involvement,
- Lightweight command and control (C2) capability representations,
- Developmental versions available for assessment,
- Minimal infrastructure,
- Flexible modeling and simulation (M&S), operator-in-the-loop (OITL), and hardware-in-the-loop (HWIL) capabilities, and
- In-line, continuous performance monitoring and selective forensics. (Roberts 2006)

Enterprise architecture (EA) can be used as a primary tool in support of evaluation and assessment. EA can be used to provide a model to understand how the parts of the enterprise fit together (or do not) (Giachetti 2010). The structure and contents of the EA should be driven by the key business decisions (or, as shown in the six-step process presented by Martin (2005), the architecture should be driven by the "business questions" to be addressed by the architecture).

The evaluation and assessment success measures can be put into the EA models and views directly and mapped to the elements that are being measured. An example of this can be seen in the US National Oceanographic and Atmospheric Agency (NOAA) EA shown by Martin (2003a and 2003b). The measures are shown, in this example, as success factors, key performance indicators, and information needs in the business strategy layer of the architecture.

EA can be viewed as either the set of artifacts developed as "views" of the enterprise, or as a set of activities that create, use, and maintain these artifacts. The literature uses these terms in both senses and it is not always clear in each case which sense is intended.

Enterprise Portfolio Considerations

Opportunity Assessment and Management

The management activities dealing with opportunities (as opposed to just risk) are included in ESE. According to White (2006), the "greatest enterprise risk may be in not pursuing enterprise opportunities." Hillson believes there is:

> *a systemic weakness in risk management as undertaken on most projects. The standard risk process is limited to dealing only with uncertainties that might have negative impact (threats). This means that risk management as currently practiced is failing to address around half of the potential uncertainties—the ones with positive impact (opportunities).* (Hillson 2004)

White claims that "in systems engineering at an enterprise scale the focus should be on opportunity, and that enterprise risk should be viewed more as something that threatens the pursuit of enterprise opportunities" (White 2006). The figure below (Rebovich and White 2011, chapter 5) shows the relative importance of opportunity and risk at the different scales of an individual system, a system of systems (SoS), and an enterprise. The implication is that, at the enterprise level, there should be more focus on opportunity management than on risk management.

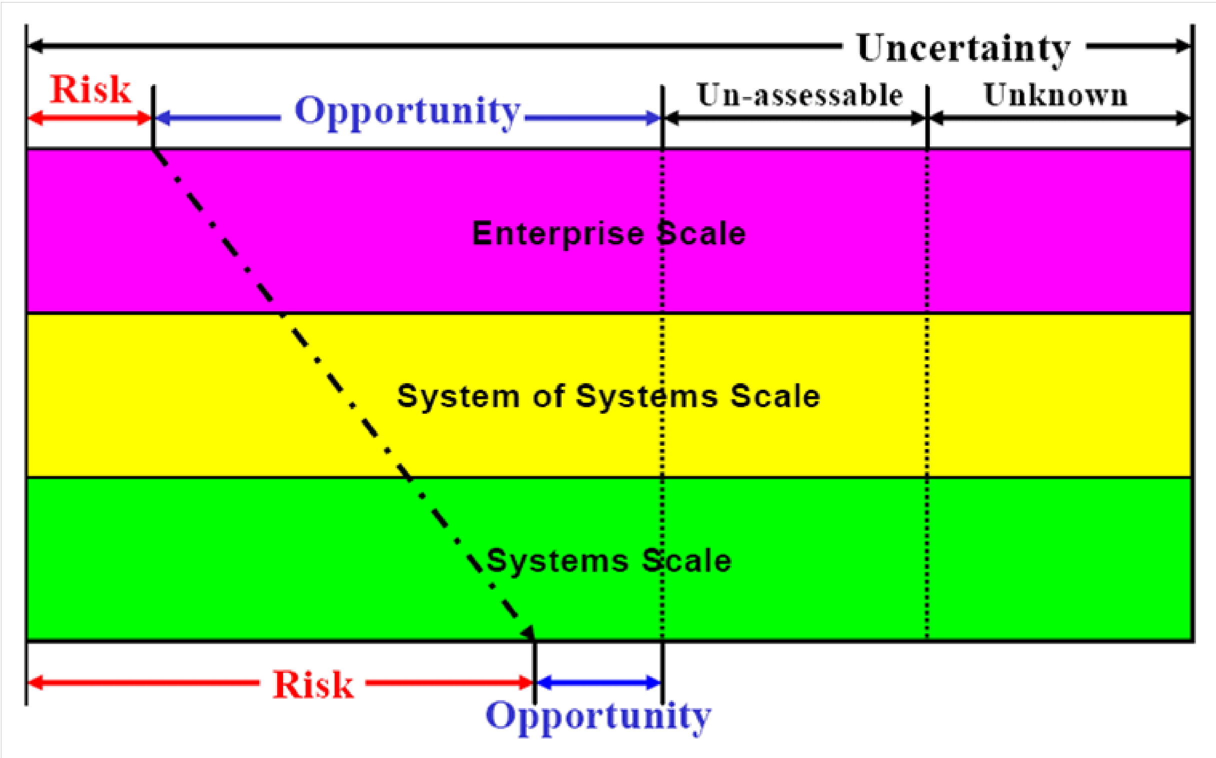


Figure 4. Risk & Opportunity at the Enterprise Scale versus the Systems Scale (White 2006). MITRE Approved for Public Release; Distribution Unlimited. Unique Tracking #05-1262.

Enterprise Architecture and Requirements

EA goes above and beyond the technical components of product systems to include additional items such as strategic goals and objectives, operators and users, organizations and other stakeholders, funding sources and methods, policies and practices, processes and procedures, facilities and platforms, infrastructure, and real estate. EA can be used to provide a model to understand how the parts of the enterprise fit together (or don't) (Giachetti 2010). The EA is not strictly the province of the chief information officer (CIO), and is not only concerned with information technology. Likewise, enterprise requirements need to focus on the cross-cutting measures necessary to ensure overall enterprise success. Some of these enterprise requirements will apply to product systems, but they may also apply to business processes, inter-organizational commitments, hiring practices, investment directions, and so on (Bernus, Nemes, and Schmidt 2003).

Architecture descriptions following the guidelines of an architecture framework have been used to standardize the views and models used in architecting efforts (Zachman 1987 and 1992; Spewak 1992). Architecture descriptions have also been developed using a business-question based approach (Martin 2003b; Martin 2006). The standard on Architecture Description Practices (ISO/IEC 42010) (ISO/IEC 2011) has expanded its scope to include requirements on architecture frameworks.

Government agencies have been increasingly turning to SE to solve some of their agency-level (i.e., enterprise) problems. This has sometimes led to the use of an architecture-based investment process, especially for information technology procurements. This approach imposes a requirement for linking business strategies to the development of EAs. The Federal Enterprise Architecture Framework (FEAF) (CIO Council 1999) and the DoD Architecture Framework (DoDAF) (DoD 2010) were developed to support such an architecture-based investment process. There have been several other architecture frameworks also developed for this purpose (ISO 2000; ISO/IEC 1998; NATO 2004; TOGAF 2009; MOD 2010; TRAK 2010).

ESE Process Elements

As a result of the synthesis outlined above, the ESE process elements to be used at the enterprise scale are as follows:

1. Strategic Technical Planning,
2. Capability-Based Planning Analysis,
3. Technology and Standards Planning,
4. Enterprise Evaluation and Assessment,
5. Opportunity and Risk Assessment and Management,
6. Enterprise Architecture and Conceptual Design,
7. Enterprise Requirements Definition and Management,
8. Program and Project Detailed Design and Implementation,
9. Program Integration and Interfaces,
10. Program Validation and Verification,
11. Portfolio and Program Deployment and Post Deployment, and
12. Portfolio and Program Life Cycle Support.

The first seven of these elements were described in some detail above. The others are more self-evident and are not discussed in this article.

References

Works Cited

Ackoff, R.L. 1989. "From Data to Wisdom." *Journal of Applied Systems Analysis*. 16 (1): 3-9.

Bernus, P., L. Nemes, and G. Schmidt (eds.). 2003. *Handbook on Enterprise Architecture*. Berlin and Heidelberg, Germany: Springer-Verlag.

CIO Council. 1999. *Federal Enterprise Architecture Framework (FEAF), Version 1.1*. Washington, DC, USA: Federal Chief Information Officers Council.

DoD. 2010. *DoD architecture framework (DoDAF)*, version 2.0. Washington, DC: US Department of Defense (DoD).

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group.

Hillson, D. 2004. *Effective Opportunity Management for Projects: Exploiting Positive Risk*. Petersfield, Hampshire, UK; New York, NY, USA: Rick Doctor & Partners; Marcel Dekker, Inc.

Hines, P., and N. Rich. 1997. "The Seven Value Stream Mapping Tools." *International Journal of Operations & Production Management*. 1 (17): 46-64.

ISO. 2000. ISO 15704:2000, *Industrial Automation Systems — Requirements for Enterprise-Reference Architectures and Methodologies.* Geneva, Switzerland: International Organization for Standardization (ISO).

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

ISO/IEC. 1998. ISO/IEC 10746:1998, *Information Technology — Open Distributed Processing — Reference Model: Architecture.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC).

Martin, J.N. 2010. "An Enterprise Systems Engineering Framework." Presented at 20th Anniversary International Council on Systems Engineering (INCOSE) International Symposium, July 12-15, 2010, Chicago, IL, USA.

Martin, J.N. 2006. "An Enterprise Architecture Process Incorporating Knowledge Modeling Methods." PhD dissertation. Fairfax, VA, USA: George Mason University.

Martin, J.N. 2005. "Using an Enterprise Architecture to Assess the Societal Benefits of Earth Science Research." Presented at 15th Annual International Council on Systems Engineering (INCOSE) International Symposium, 2005, Rochester, NY, USA.

Martin, J.N. 2003a. "An Integrated Tool Suite for the NOAA Observing System Architecture." Presented at 13th Annual International Council on Systems Engineering (INCOSE) International Symposium, 2003, Arlington, VA, USA.

Martin, J.N. 2003b. "On the Use of Knowledge Modeling Tools and Techniques to Characterize the NOAA Observing System Architecture." Presented at 13th Annual International Council on Systems Engineering (INCOSE) International Symposium, 2003, Arlington, VA, USA.

McCaughin, K., and J.K. DeRosa. 2006. "Process in Enterprise Systems Engineering." Presented at 16th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2006, Orlando, FL, USA.

MOD. 2010. *Ministry of Defence Architecture Framework (MODAF)*, version 1.2.004. London, England, UK: UK Ministry of Defence. Accessed September 8, 2011. Available: http:/ / www. mod. uk/ NR/ rdonlyres/ 04B5FB3F-8BBC-4A39-96D8-AFA05E500E4A/0/20100602MODAFDownload12004.pdf.

NATO. 2010. *NATO Architecture Framework (NAF)*, version 3.1. Brussels, Belgium: North Atlantic Treaty Organization.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, Auerbach.

Roberts, J.L. 2006. "Enterprise Analysis and Assessment." Presented at 16th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2006, Orlando, FL, USA.

Rother, M. 2009. *Toyota Kata: Managing People for Improvement, Adaptiveness, and Superior Results*. New York, NY, USA: McGraw-Hill.

Rother, M., and J. Shook. 1999. *Learning to See: Value-Stream Mapping to Create Value and Eliminate MUDA*. Cambridge, MA, USA: Lean Enterprise Institute.

Spewak, S.H. 1992. *Enterprise Architecture Planning: Developing a Blueprint for Data, Applications and Technology*. New York, NY, USA: Wiley and Sons, Inc.

TOGAF. 2009. "The Open Group Architecture Framework," version 9. Accessed September 2, 2011. Available: http://www.opengroup.org/togaf.

TRAK. 2011. "TRAK Enterprise Architecture Framework." Accessed September 7, 2011. Available: http://trak.sourceforge.net/index.html.

White, B.E. 2006. "Enterprise Opportunity and Risk." Presented at 16th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2010, Orlando, FL, USA.

Zachman, J.A. 1992. "Extending and Formalizing the Framework for Information Systems Architecture." *IBM Systems Journal*. 31 (3): 590-616.

Zachman, J.A. 1987. "A Framework for Information Systems Architectures." *IBM Systems Journal*. 26 (3): 276-292.

Primary References

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group.

Martin, J.N. 2010. "An Enterprise Systems Engineering Framework." Presented at 20th Anniversary International Council on Systems Engineering (INCOSE) International Symposium, July 12-15, 2010, Chicago, IL, USA.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, Auerbach.

Additional References

DeRosa, J.K. 2005. "Enterprise Systems Engineering." Presented at Air Force Association, Industry Day, Day 1, August 4, 2005, Danvers, MA, USA.

Holt, J., and S. Perry. 2010. *Modelling enterprise architectures*. Stevenage, England, UK: Institution of Engineering and Technology (IET).

Kaplan, R., and D. Norton. 1996. *The Balanced Scorecard: Translating Strategy into Action*. Cambridge, MA, USA: Harvard Business School Press.

McGovern, J., S. Ambler, M. Stevens, J. Linn, V. Sharan, and E. Jo. 2004. *A Practical Guide to Enterprise Architecture*. New York, NY, USA: Prentice Hall.

Swarz, R.S., J.K. DeRosa, and G. Rebovich. 2006. "An Enterprise Systems Engineering Model." INCOSE Symposium Proceedings, July 9-13, 2006, Orlando, FL, USA.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Enterprise Capability Management

Introduction

There are three different kinds of capability: organizational capability, system capability, and operational capability. Management of organizational capability is addressed in the article called Enabling Businesses and Enterprises. Management of system capability is addressed by the Systems Engineering (SE) management activities described in the articles called Systems Engineering Management and Product and Service Life Management. Management of operational capability is described herein.

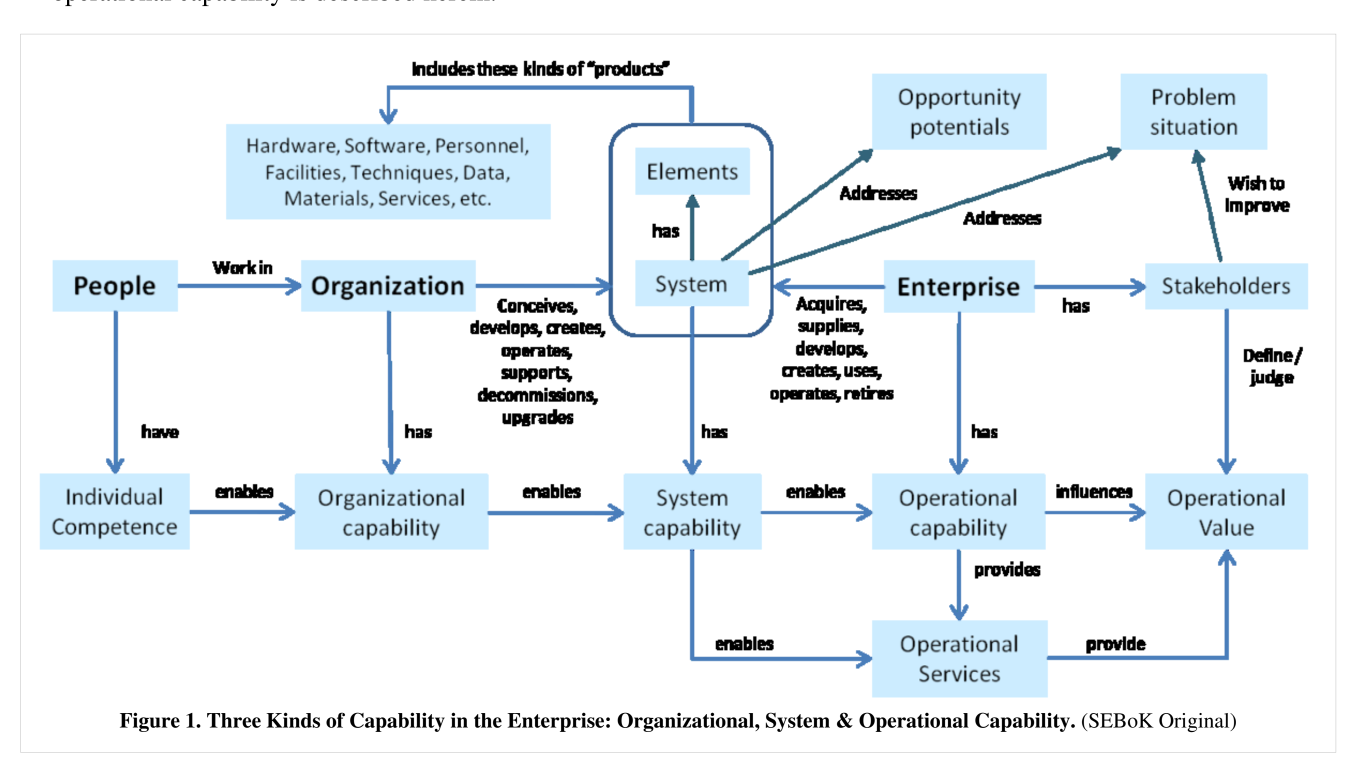


Figure 1. Three Kinds of Capability in the Enterprise: Organizational, System & Operational Capability. (SEBoK Original)

The enterprise (glossary) has a current and planned (baseline) operational capability, based on its past activities and on its current plans for change. The purpose of the enterprise capability management function is to ensure the possibility of "vectoring" the enterprise away from the current baseline trajectory to a more desirable position where it can better meet its enterprise strategic goals and objectives, given all its resource constraints and other limitations.

Operational capability may need to include elements identified in the Information Technology Infrastructure Library (ITIL) best practices for operations management, starting with strategic operation planning (OGC 2009).

> *The ITIL is a set of practices for IT service management (ITSM) that focuses on aligning IT services with the needs of business. In its current form …, ITIL is published in a series of five core publications, each of which covers an ITSM lifecycle stage.*
>
> *ITIL describes procedures, tasks and checklists that are not organization-specific, used by an organization for establishing a minimum level of competency. It allows the organization to establish a baseline from which it can plan, implement, and measure. It is used to demonstrate compliance and to measure improvement.* (http:/ / en. wikipedia. org/ wiki/ Information_Technology_Infrastructure_Library).

Needs Identification & Assessment

The enterprise has key stakeholders that have operational (glossary) needs they would like the enterprise to address. These operational needs must be identified and assessed in terms of their relevance to the enterprise and the relative priorities of these needs compared to each other and to the priorities of the enterprise itself. The enterprise exists to meet these needs. An operational need is an expression of something desirable in direct support of the enterprise's end user (glossary) activities. End user activities include such things as retail sales, entertainment, food services, and business (glossary) travel. An example of an operational need is: "Provide transportation services to commuters in the metropolitan area of London."

Enterprise needs can be much more than eliminating waste, and the challenge for ESE might relate to any or all of the following: countering a perceived threat (business or military), meeting a policy goal (as in government), doing existing business more efficiently, taking advantage of technological opportunities, meeting new operational needs, replacing obsolete systems, creating integrated enterprises with others (on a temporary or permanent basis), and so on.

In addition to operational needs, there are enterprise needs that relate to enabling assets the enterprise has in place that allow the mission to be accomplished. Enabling assets are things such as personnel, facilities, communication networks, computing facilities, policies and practices, tools and methods, funding and partnerships, equipment and supplies, and so on. An enterprise need is an expression of something desirable in direct support of the enterprise's internal activities. Internal activities include such things as market forecast, business development, product (glossary) development, manufacturing, and service delivery.

The purpose of the enterprise's enabling assets is to effect state changes to relevant elements of the enterprise necessary to achieve targeted levels of performance. The enterprise "state" shown in the figure below is a complex (glossary) web of past, current and future states (Rouse 2009). The enterprise work processes use these enabling assets to accomplish their work objectives in order to achieve the desired future states. Enterprise architecture (EA) can be used to model (glossary) these states and the relative impact each enabling asset has on the desired state changes.

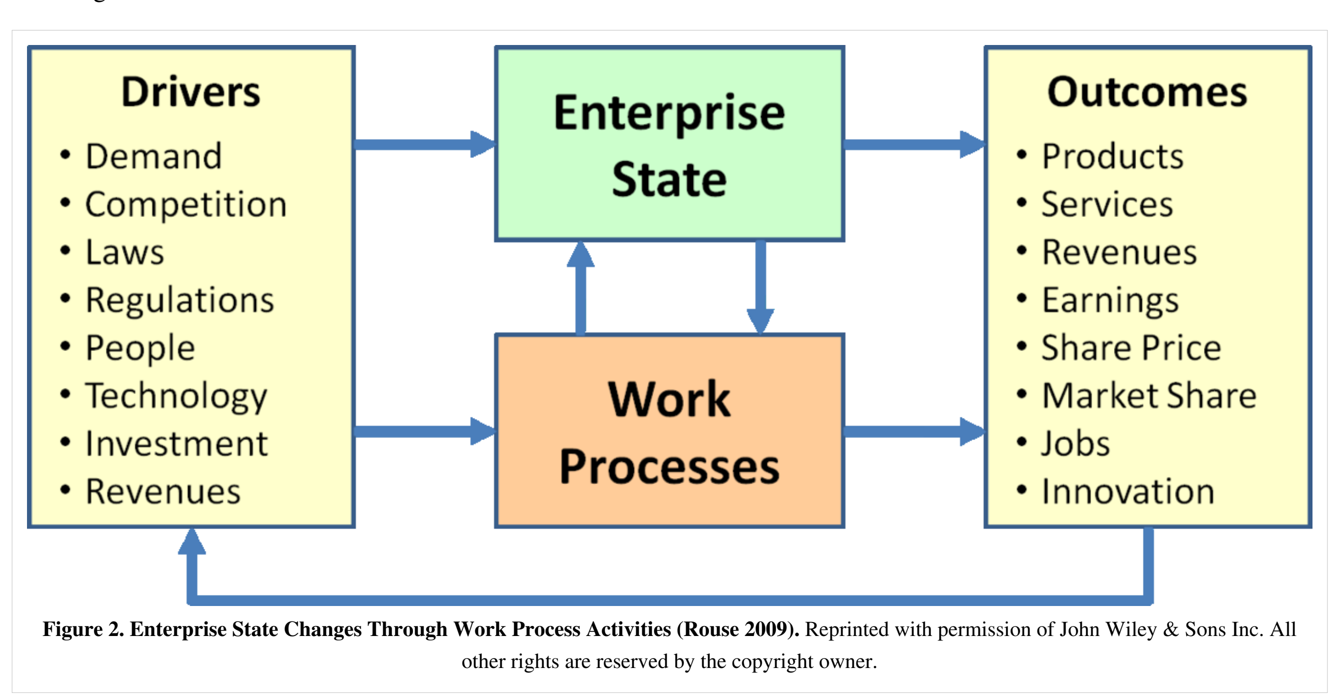


Figure 2. Enterprise State Changes Through Work Process Activities (Rouse 2009). Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Enterprise needs are related to the enterprise efficiencies achieved through the performance of enterprise activities. The main goal of enterprise needs is to maximize the efficient utilization of enterprise assets, or in other words, enhance productivity, and find and eliminate waste. Waste represents that which does not contribute to the enterprise mission or that cannot reasonably be expected to be accomplished by the enterprise. An example of an enterprise

need is: “Decrease power required for operation of enterprise data centers.” (Power is a limited asset that consumes scarce enterprise funds that could be used for delivery of other more valuable services to its customers.)

Capability Identification & Assessment

The capabilities of an enterprise should exist for the sole purpose of meeting mission and enterprise needs. Hence, there will be both mission and enterprise capabilities to identify and assess how well they meet these needs. An example of an operational capability is: “Transport 150,000 passengers per hour among 27 nodes in the network.” A supporting enterprise capability might be: “Process 200,000 tickets per hour during peak loading.” There is a baseline capability due to capability development up to that point in time, plus any additional capability planned for the future. The desired levels of capability (based on needs assessment) are compared to the baseline capability to determine the capability gaps for the enterprise. This activity will also determine points of excess capability.

The gaps should be filled and the excesses should be eliminated. The projected gaps and excesses are sometimes mapped into several future timeframes to get a better understanding of the relative timing and intensity of change that might be required. It is typical to use time “buckets” like near-term, mid-term, and far-term, which, for some long-lasting capabilities, might correspond to five, ten, and twenty years out respectively. Of course, for fast-changing capabilities (like consumer products) these timeframes would necessarily be shorter in duration, for example, one, two and three years out.

Enterprise Architecture Formulation & Assessment

Enterprise architecture analysis can be used to determine how best to fill these capability gaps and minimize the excess capabilities (or “capacities”). Usually a baseline architecture is characterized for the purpose of understanding what one currently has and where the enterprise is headed under the current business plans. The needs and gaps are used to determine where in the architecture elements need to be added, dropped, or changed. Each modification represents a potential benefit to various stakeholders, along with associated costs and risks for introducing that modification. Enterprise architecture can be used to provide a model to understand how the parts of the enterprise fit together (or do not) (Giachetti 2010).

The enterprise architecture effort supports the entire capability management activity with enterprise-wide views of strategy, priorities, plans, resources, activities, locations, facilities, products, services, and so on (ISO/IEC/IEEE 15288 (ISO/IEC/IEEE 2015) and architectural design process: ISO/IEC 42010 (ISO/IEC 2011) and ISO 15704 (ISO 2000)).

Opportunity Identification & Assessment

The enterprise architecture is used to help identify opportunities for improvement. Usually these opportunities require the investment of time, money, facilities, personnel, and so on. There might also be opportunities for “divestment,” which could involve selling of assets, reducing capacity, canceling projects, and so on. Each opportunity can be assessed on its own merits, but usually these opportunities have dependencies and interfaces with other opportunities, with the current activities and operations of the enterprise, and with the enterprise's partners. Therefore, the opportunities may need to be assessed as a “portfolio,” or, at least, as sets of related opportunities. Typically, a business case assessment is required for each opportunity or set of opportunities.

Enterprise Portfolio Management

If the set of opportunities is large or has complicated relationships, it may be necessary to employ portfolio management techniques. The portfolio elements could be bids, projects, products, services, technologies, intellectual property, etc., or any combination of these items. Examples of an enterprise portfolio captured in an architecture modeling tool can be found in Martin (2005), Martin et al. (2004), and Martin (2003). See Kaplan's work (2009) for more information on portfolio management, and ISO/IEC (2008) for information on projects portfolio management process.

Enterprise Improvement Planning & Execution

The results of the opportunity assessment are compiled and laid out in an enterprise plan that considers all relevant factors, including system capabilities, organizational capabilities, funding constraints, legal commitments and obligations, partner arrangements, intellectual property ownership, personnel development and retention, and so on. The plan usually goes out to some long horizon, typically more than a decade, depending on the nature of the enterprise's business environment, technology volatility, market intensity, and so on. The enterprise plan needs to be in alignment with the enterprise's strategic goals and objectives and with leadership priorities.

The planned improvements are implemented across the enterprise and in parts of the extended enterprise (glossary) where appropriate, such as suppliers in the supply chain, distributors in the distribution chain, financiers in the investment arena, and so on. The planned changes should have associated performance targets and these metrics should be monitored to ensure that progress is being made against the plan and that the intended improvements are being implemented. As necessary, the plan is adjusted to account for unforeseen circumstances and outcomes. Performance management of enterprise personnel is a key element of the improvement efforts.

Enterprise Capability Change Management

In an operational context (glossary) (particularly in defense) the term "capability management" is associated with developing and maintaining all aspects of the ability to conduct certain types of missions in a given threat (glossary) environment. In an industrial context, capability refers to the ability to manage certain classes of product and service through those parts of their life cycle that are relevant to the business. Changes to enterprise capability should be carefully managed to ensure that current operations are not adversely affected (where possible) and that the long term viability of the enterprise is maintained. The following seven lenses can be used to facilitate change management: strategic objectives, stakeholders, processes, performance metrics, current state alignment, resources, and maturity assessment (Nightingale and Srinivasan 2011).

Capability management is becoming more often recognized as a key component of the business management tool suite:

> *Capability management aims to balance economy in meeting current operational requirements, with the sustainable use of current capabilities, and the development of future capabilities, to meet the sometimes competing strategic and current operational objectives of an enterprise. Accordingly, effective capability management assists organizations to better understand, and effectively integrate, re-align and apply the total enterprise ability or capacity to achieve strategic and current operational objectives; and develops and provides innovative solutions that focus on the holistic management of the defined array of interlinking functions and activities in the enterprise's strategic and current operational contexts.* (Saxena 2009, 1)

There is a widespread perception that capability management is only relevant to defense and aerospace domains. However, it is becoming more widely recognized as key to commercial and civil government efforts.

References

Works Cited

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group.

ISO. 2000. ISO 15704:2000, *Industrial Automation Systems — Requirements for Enterprise — Reference Architectures and Methodologies.* Geneva, Switzerland: International Organization for Standardization (ISO).

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

Kaplan, J. 2009. *Strategic IT Portfolio Management: Governing Enterprise Transformation*. Waltham, MA, USA: Pittiglio, Rabin, Todd & McGrath, Inc. (PRTM).

Martin, J.N. 2005. "Using an Enterprise Architecture to Assess the Societal Benefits of Earth Science Research." Presented at 15th Annual International Council on Systems Engineering (INCOSE) International Symposium, 2005, Rochester, NY, USA.

Martin, J.N. 2003. "On the Use of Knowledge Modeling Tools and Techniques to Characterize the NOAA Observing System Architecture." Presented at 13th Annual International Council on Systems Engineering (INCOSE) International Symposium, 2003, Arlington, VA, USA.

Martin, J.N., J. Conklin, J. Evans, C. Robinson, L. Doggrell, and J. Diehl. 2004. "The Capability Integration Framework: A New Way of doing Enterprise Architecture." Presented at 14th Annual International Council on Systems Engineering (INCOSE) International Symposium, June 20-24, 2004, Toulouse, France.

Nightingale, D., and J. Srinivasan. 2011. *Beyond the Lean Revolution: Achieving Successful and Sustainable Enterprise Transformation*. New York, NY, USA: AMACOM Press.

OGC (Office of Government Commerce). 2009. *ITIL Lifecycle Publication Suite Books*. London, UK: The Stationery Office.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of Systems Engineering and Management*, 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY, USA: Wiley and Sons, Inc.

Saxena, M.S. 2009. *Capability Management: Monitoring & Improving Capabilities*. New Dehli: Global India Publications Pvt Ltd.

Wikipedia contributors. "Information Technology Infrastructure Library." *Wikipedia, The Free Encyclopedia*. Accessed November 28, 2012. Available at: http:/ / en. wikipedia. org/ wiki/ Information_Technology_Infrastructure_Library.

Primary References

Kaplan, J. 2009. *Strategic IT Portfolio Management: Governing Enterprise Transformation*. Waltham, MA, USA: Pittiglio, Rabin, Todd & McGrath, Inc. (PRTM).

Nightingale, D., and J. Srinivasan. 2011. *Beyond the Lean Revolution: Achieving Successful and Sustainable Enterprise Transformation*. New York, NY, USA: AMACOM Press.

Rouse, W.B. 2009. "Engineering the Enterprise as a System," in *Handbook of Systems Engineering and Management*, 2nd ed., edited by A.P. Sage and W.B. Rouse. New York, NY, USA: Wiley and Sons, Inc.

Additional References

Dahmann, J.S., J.A. Lane, and G. Rebovich. 2008. "Systems Engineering for Capabilities." *CROSSTALK: The Journal of Defense Software Engineering* 21 (11): 4-9.

Hillson, D. 2004. *Effective Opportunity Management for Projects: Exploiting Positive Risk*. Petersfield, Hampshire, UK; New York, NY: Rick Doctor & Partners; Marcel Dekker, Inc.

Lillehagen, F., J. Kostie, S. Inella, H.G. Solheim, and D. Karlsen. 2003. "From enterprise modeling to enterprise visual scenes." Presented at International Society for Pharmaceutical Engineering (ISPE) Conference on Concurrent Engineering (CE), July 26-30, 2003, Madeira Island, Portugal.

McGovern, J., S. Ambler, M. Stevens, J. Linn, V. Sharan, and E. Jo. 2004. *A Practical Guide to Enterprise Architecture*. New York, NY: Prentice Hall.

Rechtin, E. 1999. *Systems Architecting of Organizations: Why Eagles Can't Swim*. Boca Raton, FL, USA: CRC Press, Taylor and Francis Group.

Roberts, J.L. 2006. "Enterprise Analysis and Assessment." Presented at 16th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2006, Orlando, FL, USA.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: System of Systems (SoS)

Systems of Systems (SoS)

System of systems engineering (SoSE) is not a new discipline; however, this is an opportunity for the systems engineering community to define the complex systems of the twenty-first century (Jamshidi 2009). While systems engineering is a fairly established field, SoSE represents a challenge for the present systems engineers on a global level. In general, SoSE requires considerations beyond those usually associated with engineering to include socio-technical and sometimes socio-economic phenomena.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Architecting Approaches for Systems of Systems
- Socio-Technical Features of Systems of Systems
- Capability Engineering

Definition and Characteristics of Systems of Systems

There are several definitions of system(s) of systems (SoS), some of which are dependent on the particularity of an application area. Maier (1998) postulated five key characteristics (not criteria) of SoS: operational independence of component systems, managerial independence of component systems, geographical distribution, emergent behavior, and evolutionary development processes, and identified operational independence and managerial independence as the two principal distinguishing characteristics for applying the term 'systems-of-systems.' A system that does not exhibit these two characteristics is not considered a system-of-systems regardless of the complexity or geographic distribution of its components.

In the Maier characterization, emergence is noted as a common characteristic of SoS particularly in SoS composed of multiple large existing systems, based on the challenge (in time and resources) of subjecting all possible logical threads across the myriad functions, capabilities, and data of the systems in an SoS. As introduced in the article Emergence, there are risks associated with unexpected or unintended behavior resulting from combining systems that have individually complex behavior. These become serious in cases which safety, for example, is threatened through unintended interactions among the functions provided by multiple constituent systems in a SoS.

ISO/IEC/IEEE 15288 Annex G (ISO, 2015) provides a definition of SoS:

> **System of Systems (SoS) —** *A system of systems (SoS) brings together a set of systems for a task that none of the systems can accomplish on its own. Each constituent system keeps its own management, goals, and resources while coordinating within the SoS and adapting to meet SoS goals.*

It should be noted that according to this definition, formation of a SoS is not necessarily a permanent phenomenon, but rather a matter of necessity for integrating and networking systems in a coordinated way for specific goals such as robustness, cost, efficiency, etc.

ISO/IEC/IEEE 15288 Annex G also describes the impact of these characteristics on the implementation of systems engineering processes. Because of the independence of the constituent systems, these processes are in most cases implemented for engineering both the systems and the system of systems, and need to be tailored to support the characteristics of SoS. These processes are shown in the table below highlighting the fact that these processes are

implemented at both the system and SoS levels, with SoSE often constrained by the systems.

Table 1. Differences Between Systems and Systems of Systems as They Apply to Systems Engineering.

SE Process	Implementation as Applied to SoS
Agreement processes	Because there is often no top level SoS authority, effective agreements among the systems in the SoS are key to successful SoSE.
Organizational project enabling processes	SoSE develops and maintains those processes which are critical for the SoS within the constraints of the system level processes.
Technical management processes	SoSE implements technical management processes applied to the particular considerations of SoS engineering - planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into a system-of-systems capability while systems continue to be responsible for technical management of their systems.
Technical processes	SoSE technical processes define the cross-cutting SoS capability, through SoS level business/mission analysis and stakeholder needs and requirements definition. SoS architecture and design frame the planning, organization and integration of the constituent systems, constrained by system architectures. Development, integration, verification, transition and validation are implemented by the systems. with SoSE monitoring and review. SoSE integration, verification, transition and validation applies when constituent systems are integrated into the SoS and performance is verified and validated.

Finally, based on work done by the INCOSE Systems of Systems Work Group (Dahmann, 2014), the major challenges facing SoSE have been catalogued in terms of seven pain points. These challenges are presented in the SoSE section of the INCOSE SE Handbook. (INCOSE 2015). These challenges include:

- **SoS Authorities**. In a SoS each constituent system has its own local ‘owner’ with its stakeholders, users, business processes and development approach. As a result, the type of organizational structure assumed for most traditional systems engineering under a single authority responsible for the entire system is absent from most SoS. In a SoS, SE relies on cross-cutting analysis and on composition and integration of constituent systems which, in turn, depend on an agreed common purpose and motivation for these systems to work together towards collective objectives which may or may not coincide with those of the individual constituent systems.
- **Leadership**. Recognizing that the lack of common authorities and funding pose challenges for SoS, a related issue is the challenge of leadership in the multiple organizational environment of a SoS. This question of leadership is experienced where a lack of structured control normally present in SE of systems requires alternatives to provide coherence and direction, such as influence and incentives.
- **Constituent Systems’ Perspectives.** Systems of systems are typically comprised, at least in part, of in-service systems, which were often developed for other purposes and are now being leveraged to meet a new or different application with new objectives. This is the basis for a major issue facing SoS SE; that is, how to technically address issues which arise from the fact that the systems identified for the SoS may be limited in the degree to which they can support the SoS. These limitations may affect the initial efforts at incorporating a system into a SoS, and systems ‘commitments to other users may mean that they may not be compatible with the SoS over time. Further, because the systems were developed and operate in different situations, there is a risk that there could be a mismatch in understanding the services or data provided by one system to the SoS if the particular system’s context differs from that of the SoS.
- **Capabilities and Requirements.** Traditionally (and ideally) the SE process begins with a clear, complete set of user requirements and provides a disciplined approach to develop a system to meet these requirements. Typically, SoS are comprised of multiple independent systems with their own requirements, working towards broader capability objectives. In the best case the SoS capability needs are met by the constituent systems as they meet their own local requirements. However, in many cases the SoS needs may not be consistent with the requirements for the constituent systems. In these cases, the SoS SE needs to identify alternative approaches to meeting those needs through changes to the constituent systems or additions of other systems to the SoS. In effect this is asking

the systems to take on new requirements with the SoS acting as the 'user'.
- **Autonomy, Interdependencies and Emergence.** The independence of constituent systems in a SoS is the source of a number of technical issues facing SE of SoS. The fact that a constituent system may continue to change independently of the SoS, along with interdependencies between that constituent system and other constituent systems, add to the complexity of the SoS and further challenges SE at the SoS level. In particular, these dynamics can lead to unanticipated effects at the SoS level leading to unexpected or unpredictable behavior in a SoS even if the behavior of constituent systems is well understood.
- **Testing, Validation, and Learning.** The fact that SoS are typically composed of constituent systems which are independent of the SoS poses challenges in conducting end-to-end SoS testing as is typically done with systems. Firstly, unless there is a clear understanding of the SoS-level expectations and measures of these expectations, it can be very difficult to assess level of performance as the basis for determining areas which need attention, or to assure users of the capabilities and limitations of the SoS. Even when there is a clear understanding of SoS objectives and metrics, testing in a traditional sense can be difficult. Depending on the SoS context, there may not be funding or authority for SoS testing. Often the development cycles of the constituent systems are tied to the needs of their owners and original ongoing user base. With multiple constituent systems subject to asynchronous development cycles, finding ways to conduct traditional end-to-end testing across the SoS can be difficult if not impossible. In addition, many SoS are large and diverse making traditional full end-to-end testing with every change in a constituent system prohibitively costly. Often the only way to get a good measure of SoS performance is from data collected from actual operations or through estimates based on modeling, simulation and analysis. Nonetheless the SoS SE team needs to enable continuity of operation and performance of the SoS despite these challenges.
- **SoS Principles.** SoS is a relatively new area, with the result that there has been limited attention given to ways to extend systems thinking to the issues particular to SoS. Work is needed to identify and articulate the cross cutting principles that apply to SoS in general, and to developing working examples of the application of these principles. There is a major learning curve for the average systems engineer moving to a SoS environment, and a problem with SoS knowledge transfer within or across organizations.

Types of SoS

In today's interconnected world, SoS occur in a broad range of circumstances. In those situations where the SoS is recognized and treated as a system in its right, an SoS can be described as one of four types (Maier 1998; Dahmann and Baldwin 2008):

- **Directed** - The SoS is created and managed to fulfill specific purposes and the constituent systems are subordinated to the SoS. The component systems maintain an ability to operate independently; however, their normal operational mode is subordinated to the central managed purpose;
- **Acknowledged** - The SoS has recognized objectives, a designated manager, and resources for the SoS; however, the constituent systems retain their independent ownership, objectives, funding, and development and sustainment approaches. Changes in the systems are based on cooperative agreements between the SoS and the system;
- **Collaborative** - The component systems interact more or less voluntarily to fulfill agreed upon central purposes. The central players collectively decide how to provide or deny service, thereby providing some means of enforcing and maintaining standards; and
- **Virtual** - The SoS lacks a central management authority and a centrally agreed upon purpose for the SoS. Large-scale behavior emerges—and may be desirable—but this type of SoS must rely on relatively invisible mechanisms to maintain it.

This taxonomy is based on the degree of independence of constituents and it offers a framework for understanding SoS based on the origin of the SoS objectives and the relationships among the stakeholders for both the SoS and its constituent systems. In most actual cases, an SoS will reflect a combination of SoS types. This taxonomy is in

general use. Other taxonomies may focus on nature/type of components, their heterogeneity, etc. (Cook 2014)

As noted above, many SoS exist in an unrecognized state; this is increasingly true as the levels of interconnectivity between modern systems keeps increasing. Kemp et al (2013) describe such systems as “accidental” but they can be described as “discovered” because it is only when they become significant for some reason that we recognize them, at which point they can usually fall into one of the above four categories, since their significance means they must now operate, with management, in some defined way.

From the SoSE point of view, another potential classification would consider the level of anticipation/preparation of SoSE with respect to SoS operations and level of stability of the SoS objectives; this is referred to as variability by kinder et. al. (2012). This could range from an SoS which responds to a particular trigger and is put immediately in place when needs are expressed. An example of such an SoS would be a crisis management SoS. This type of SoS is updated dynamically during the operation. At the other end of the spectrum there are well-specified and stable SoS developed to answer to specified ongoing needs. An example of such a persistent SoS is an air traffic management system. This type of SoS is acquired and qualified in a well-defined environment and any need for evolution will imply a formal SE evolution and re-qualification.

SoSE Application Domains

Application of SoSE is broad and is expanding into almost all walks of life. Originally identified in the defense environment, SoSE application is now much broader and still expanding. The early work in the defense sector has provided the initial basis for SoSE, including its intellectual foundation, technical approaches, and practical experience. In addition, parallel developments in information services and rail have helped to develop SoSE practice (Kemp and Daw, 2015). Now, SoSE concepts and principles apply across other governmental, civil and commercial domains.

Some examples include:

- **Transportation** - air traffic management, the European rail network, integrated ground transportation, cargo transport, highway management, and space systems,
- **Energy** - smart grid, smart houses, and integrated production/consumption,
- **Health Care** - regional facilities management, emergency services, and personal health management,
- **Defense** - Military missions such as missile defense, networked sensors,
- **Rail** – Urban, national, international rail systems,
- **Natural Resource Management** - global environment, regional water resources, forestry, and recreational resources,
- **Disaster Response** - responses to disaster events including forest fires, floods, and terrorist attacks,
- **Consumer Products** - integrated entertainment and household product integration,
- **Business-** banking and finance, and
- **Media** - film, radio, and television.

Increased networking and interconnectedness of systems today contributes to growth in the number and domains where SoS are becoming the norm, particularly with the considerable converge among systems of systems, cyber-physical systems and the internet of things. (Henshaw, 2016).

Difference between System of Systems Engineering and Systems Engineering

Observations regarding differences between individual or constituent systems and SoS are listed in Table 1. These differences are not as black and white as the table might suggest and in each case, the degree of difference varies in practice. Modern systems tend to be highly inter-connected, so that the assumptions that lead to the characteristics of Systems Engineering in Table 2 are less frequently met.

Table 2. Differences Between Systems and Systems of Systems as They Apply to Systems Engineering. (SEBoK Original), adapted from Dahmann and Baldwin (2008) and Neaga et al. (2009)

	Systems Engineering	Systems of Systems Engineering
Management and Oversight		
Stakeholder Involvement	Clear set of stakeholders	Multiple levels of stakeholders with mixed and possibly competing interests
Governance	Aligned management and funding	Added levels of complexity due to management and funding for both SoS and systems; SoS does not have control over all constituent systems
Operational Focus (Goals)		
Operational Focus	Designed and developed to meet common objectives	Called upon to meet new SoS objectives using systems whose objectives may or may not align with the SoS objectives
Implementation		
Acquisition/Development	Aligned to established acquisition and development processes	Cross multiple system lifecycles across asynchronous acquisition and development efforts, involving legacy systems, developmental systems, and technology insertion
Process	Well-established	Learning and adaptation
Test and Evaluation	Test and evaluation of the system is possible System requirements drive the system T&E and use Measures of Performance (MoP)	Testing is more challenging due to systems' asynchronous life cycles of component systems and the complexity of all the parts. At the SoS level, Measures of Effectiveness are needed, which are difficult to define, as well as MoPs
Engineering and Design Considerations		
Boundaries and Interfaces	System of Interest (SOI) is defined by focusing on boundaries and interfaces	The dynamic and reconfigurable nature of SoS mean that boundaries and interfaces may change. Also, component systems may belong to more than one SoS and have variable availability
Performance and Behavior	Performance of the system to meet performance objectives	Performance across the SoS that satisfies SoS capability objectives while balancing needs of the constituent systems
Metrics	Derivation from requirements is straightforward	Difficult to define, agree, and quantify due to independent management of component systems

Standards

The study concluded (ISO,2015): Standards for system of systems engineering beginning to emerge as the practice of SoSE matures. As a recent report of an ISO SoS Standards study group states, “A review of existing standards identified only a few places where SoS is explicitly addressed by current standards. In IS/IEC JTC1 SC7, the one place where SoS is explicitly addressed is in ISO/IEC/IEEE 15288:2015: Systems and Software Engineering Lifecycle Processes. Annex G to 15288 focuses on a description of how the processes in the body of 15288 are adapted for use with SoS. The driving characteristics and types of SoS are described as are the impact these have on the implementation of the life cycle processes.” (ISO, 2015)

Based on the material in the preceding sections, the study has demonstrated that there is:

- A strong foundation for SoSE in fundamentals and continued growth in emerging areas
- SoSE activity across multiple domains and growing demand also across SoS domains
- Limited cross domain communication among practitioners
- Limited understanding and appreciation of current SoSE knowledge base, including current practice and research

In terms of SoS standards:

- The only SoS-specific standards are those in Annex G to ISO/IEC/IEEE 15288:2015
- There are no top level standards to aid in communication across SoSE practitioners and domains
- There is no guidance on how SoSE can employ existing standards (and not generate new duplicative standards).

Based on the study results, work has been initiated on a set of SoSE standards development activities to address this gap.

References

Works Cited

Cook, S. C. and Pratt, J. M., “Towards designing innovative SoSE approaches for the Australian defence force,” Proc. 9th Int. Conf. Syst. Syst. Eng. Socio-Technical Perspect. SoSE 2014, pp. 295–300, 2014.

Dahmann, J., and K. Baldwin. 2008. "Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering." Presented at IEEE Systems Conference, April 7-10, 2008, Montreal, Canada.

DoD. 2008. *Systems Engineering Guide for Systems of Systems.* Arlington, VA: US Department of Defense, Director, Systems and Software Engineering, Deputy Under Secretary of Defense (Acquisition and Technology), Office of the Under Secretary of Defense (Acquisition, Technology and Logistics). Accessed November 12, 2013. Available: http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

Henshaw, M.J.d,, “Systems of Systems. Cyber-Physical Systems, the Internet of Things… Whatever Next?” INCOSE INSIGHT, October 2016, Volume 19, Issue 3, pages 51-54.

International Standards Organization, 2016. Report of the SC7 SG on Systems of Systems Engineering.

Kemp, D., et. al.. 2013. *Steampunk System of Systems Engineering: A case study of successful System of Systems engineering in 19th century Britain*." Presented at INCOSE International Symposium, June 24–27, 2013, Philadelphia, PA.

Kinder, A., Barot, V., Henshaw, M., & Siemieniuch, C. (2012). System of Systems: “Defining the system of interest.” In Proc. 7th Int. Conf. Systems of Systems Eng., Genoa, italy (pp. 463–468). Retrieved from http:// ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6384211

Neaga, E.I., M.J.d. Henshaw, and Y. Yue. 2009. "The influence of the concept of capability-based management on the development of the systems engineering discipline." *Proceedings of the 7th Annual Conference on Systems Engineering Research,* April 20-23, 2009, Loughborough University, Loughborough, England, UK.

Maier, M.W. 1998. "Architecting Principles for Systems-of-Systems." *Systems Engineering.* 1 (4): 267-284.

Primary References

Dahmann, J., and K. Baldwin. 2008. "Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering." Presented at IEEE Systems Conference, April 7-10, 2008, Montreal, Canada.

DoD. 2008. Systems Engineering Guide for Systems of Systems, version 1.0. Washington, DC, USA: US Department of Defense (DoD). Available: http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

INCOSE Systems Engineering Handbook, 4th Edition, John Wiley & Sons Inc., 2015

International Standards Organization, 2016. Report of the SC7 SG on Systems of Systems Engineering.

Jamshidi, M. (ed). 2009a. *Systems of Systems Engineering – Innovations for the 21st Century*. Hoboken, NJ, USA: Wiley.

Maier, M.W. 1998. "Architecting Principles for Systems-of-Systems." *Systems Engineering*. 1 (4): 267-284.

Additional References

Barot, V., S. Henson, M. Henshaw, C. Siemieniuch, M. Sinclair, S.L. Lim, M. Jamshidi, and D. DeLaurentis. 2012. *Trans-Atlantic Research and Education Agenda in Systems of Systems (T-AREA-SoS) SOA Report*. Longborough, England, UK: Longborough University. Ref. TAREA-RE-WP2-R-LU-7.

Boardman, J., and B. Sauser. 2006. "System of Systems - the Meaning of Of." IEEE Conference on Systems of Systems Engineering, April 24-26, 2006, Los Angeles, CA.

Carlock, P., and J.A. Lane. 2006. *System of Systems Enterprise Systems Engineering, the Enterprise Architecture Management Framework, and System of Systems Cost Estimation*. Los Angeles, CA, USA: Center for Systems and Software Engineering (CSSE), University of Southern California (USC). USC-CSE-2006-618.

Checkland, P.B. 1999. *Systems Thinking, Systems Practice*. Chichester, UK: John Wiley & Sons Ltd.

Dahmann, J., Rebovich, G., Lane, J., Lowry, R. & Baldwin, K. 2011. "An Implementer's View of Systems Engineering for Systems of Systems." IEEE Systems Conference, April 4-7, 2011, Montreal, Canada. p. 212-217.

Keating C.B., J.J. Padilla, and K. Adams. 2008. "System of systems engineering requirements: Challenges and guidelines". *EMJ - Engineering Management Journal.* 20 (4): 24-31.

Luzeaux, D., and J.R. Ruault. 2010. *Systems of Systems*. London, UK: ISTE.

Poza, A.S., S. Kovacic, and C. Keating. 2008. "System of Systems Engineering: An Emerging Multidiscipline". *International Journal of System of Systems Engineering.* 1 (1/2).

Rebovich Jr., G. 2009. "Chapter 6: Enterprise System of Systems," in *Systems of Systems Engineering - Principles and Applications*. Boca Raton, FL, USA: CRC Press.

Ring J. 2002. "Toward an ontology of systems engineering." *INSIGHT.* 5 (1): 19-22.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Architecting Approaches for Systems of Systems

A key part of systems engineering (SE) for system of systems (SoS) is the composition of systems to meet SoS needs. This may include simply interfacing systems and leveraging their existing functionality or it may require changing the systems functionality, performance or interfaces. These changes occur incrementally, as a SoS evolves over time to meet changing SoS objectives. System of Systems Engineering (SoSE) supports these changes by developing and evolving a technical framework that acts as an overlay to the systems of which the SoS is composed. This framework provides the *architecture* for the SoS. The SoS architecture defines how the systems work together to meet SoS objectives and considers the details of the individual systems and their impact the SoS performance or functionality.

The Role of System of Systems Architecting

An architecture is the structure of components, their relationships, and the principles and guidelines governing their design evolution over time (IEEE 610.12-1990).

In a SoS, the architecture is the technical framework for the systems comprising the SoS which designates how the systems will be employed by the users in an operational setting (sometimes called the concept of operations (CONOPs or CONOPs), the internal and external relationships and dependencies among the constituent systems and their functions and, finally, the end-to-end functionality and data flow as well as communications among the systems.

Because SoS largely comprise extant independent systems, these place constraints on the SoS architecture and may require that migration to a SoS architecture be incremental. Developing a SoS architecture requires consideration of technical feasibility for the constituent systems as well as the needs of the SoS itself. Architecture data for the constituent systems can also be important data for architecting the SoS. There is some similarity here to the introduction of Commercial Off The Shelf (COTS) products into systems: the COTS product has been independently managed but sufficient data is required by the systems developer to ensure satisfactory integration. However, in this case the COTS product is not independently operated

Maier (1998) provides a conceptual discussion on the impact of SoS characteristics on SoS architecting. Additionally, the US DoD SE Guide for SoS (2008) describes practical considerations in developing and evolving a SoS architecture as a core element of SoSE.

Challenges in Architecting SoS

In the case of a new system development, the systems engineer can begin with a fresh, unencumbered approach to architecture. However, in a SoS, the systems contributing to the SoS objectives are typically in place when the SoS is established and the SoS engineer needs to consider the current state and plans of the individual systems as factors in developing an architecture for the SoS. This, along with the fact that constituent systems may be complex systems in their own right, leads to a set of challenges to architecting SoS. The approach to architecting must be determined by the type of SoS under consideration. Whereas a directed SoS can be architected in much the same way as a monolithic system, the other types are less straightforward, because the SOI may be less clearly defined and because the SoS architects knowledge of the constituent systems may be partial. Furthermore, whereas in a directed SoS the owner may have authority and funding to require architectural changes of constituent systems, in acknowledged and collaborative SoS re-architecting is at the discretion of the owners of the constituent systems. Maier (Maier 1998) has focused architecting attention on communication for SoS, arguing that this is the common feature of all types, and he partitions the communication into layers that have a similarity to the layers of interoperability (NCOIC, 2008).

The independence of the constituent systems means that these systems are typically not designed to optimize SoS objectives. It may even be the case that a constituent system should operate sub-optimally at the system level in order to achieve overall SoS effectiveness. (Rebovich 2009) has articulated this difficulty as a fundamental problem of SoS:

> *From the single-system community's perspective, its part of the SoS capability represents additional obligations, constraints and complexities. Rarely is participation in an (sic) SoS seen as a net gain from the viewpoint of single-system stakeholders.*

The development and implementation of a SoS architecture may be significantly constrained by a reluctance to make changes or invest in the constituent systems, which could be very mature (e.g. in sustainment) or currently productively supporting other uses. In this case, approaches such as gateways and wrapping may be used to incorporate these systems into the SoS without making significant changes in the other systems.

Architecture Analysis

Large-scale systems integration has grown in importance and correspondingly, there has been a growing interest in SoS concepts and strategies. The performance and functioning of groups of heterogeneous systems has become the focus of various applications including military, security, aerospace, distributed energy, healthcare, and disaster management systems (Lopez 2006; Wojcik and Hoffman 2006). There is an increasing interest in exploiting synergy between these independent systems to achieve the desired overall system performance (Azarnoush et al. 2006).

Modeling and simulation is conducted to analyze architecture effectiveness and to verify architectural features. In the literature, researchers have addressed the issues of coordination and interoperability in a SoS (Abel and Sukkarieh 2006; Sahin et al. 2007). In order to study SoS characteristics and parameters, one needs to have realistic simulation frameworks properly designed for system of systems architecture. There are some attempts to develop simulation frameworks for multi-agent systems using Discrete Event Simulation (DEVS) tools (Zeigler et al. 2000a). In these research efforts, the major focus is given to DEVS architecture with JAVA. In (Mittall 2000), DEVS state machine approach is introduced. Finally, DEVS Modeling Language (DEVSML) is developed by using XML based JAVA in order to simulate systems in a net-centric way with relative ease. Sahin et al. (2007) have recently introduced a discrete event XML based SoS simulation framework based on DEVS and JAVA.

The Open Approach to SoS Engineering

As noted above, one of the key challenges with SoS architecting is that the constituent systems of a SoS may not have been designed, developed and employed with regard to their role in the SoS, which constrains SoS architecture options. The degree to the architecture which *overlays* these constituent systems and supports the SoS end-to-end capabilities can be based on open standards; the SoS may be able to benefit from open architecture for future evolution.

The critical challenge of moving from SoS, as a concept to the engineering of SoS, is the significant technological, human, and organizational differences in consideration system of systems engineering and management approaches (Wells and Sage 2008). A potential approach to engineering a SoS can be the *open systems approach* to SoSE (Azani 2009). The following open systems principles are listed by Azani (2009):

- **Open interface principle** - Open systems have permeable boundaries that allow them to exchange mass, energy, and information with other systems;
- **Synergism principle** – The notion that designates that the co-operative interaction between constituent systems has a greater effect in their combined efforts than the sum of their individual parts. Essentially, this is what gives rise to emergence;
- **Self-government principle** - This implies that the SoS maintains and develops its internal order without interference from external sources. This could be through cybernetic control, homeostasis, or self-organization;

- **Emergence principle** - In this case, this refers to the occurrence of novel and coherent structures, patterns, and properties during the self-organization of the SoS;
- **Conservation principle** – This principle states that energy and mass (material) are conserved within the SoS;
- **Reconfiguration principle** – This refers to the SoS reconfiguring and adapting itself to sustain itself against changes in its environment;
- **Symbiosis principle** - The systems within the SoS have a symbiotic relationship to each other; more transparently, the successful development and sustainment of a SoS depends on symbiotic collaboration between the stakeholders of the systems of which it is comprised; and
- **Modularity principle** - This holds that each level and each system is to some extent independent of others. In SoS design, the development of independent modular systems that interoperate with each other through standardized interfaces enables greater flexibility to promote better evolution of the SoS.

Azani (2009) elaborates on the open systems development strategies and principles utilized by biotic SoS, discusses the implications of engineering of man-made SoS, and introduces an integrated SoS development methodology for the engineering and development of an adaptable, sustainable, and interoperable SoS based on open systems principles and strategies.

Hitchens (2003, 107), on the other hand, discusses the principles of open systems rather differently in terms of their systems life cycles, as the seven principles that he addresses are system reactions, system cohesion, system adaptation, connected variety, limited variety, preferred patterns, and cyclic progression. This description takes a systems dynamics approach to show how open systems evolve; the description is applicable to natural and man-made systems.

The enablers of openness include open standards and open specifications, which draw from consensus amongst a community of interest, and are published by, and freely available within, that community. An open specification must ensure that its detail-level is allows for it to be implementable by independent parties. Compliance with open standards is intended to ensure consistent results (Henshaw, et. al., 2011). This parallels the notion of open systems architecture, which is an open specification of the architecture of a system or system of systems for the purpose of acquiring specified capabilities. As a general feature of good design (for a system or system of systems), an open system architecture should allow for the quick and easy improvement and updating of the system capabilities, by adding or changing components. However, Henshaw et. al. (2011) also denote that open architecture represents a commercial challenge (rather than a technical one) and that establishing open architecture approaches to acquisition can be challenging, due to issues involving protection of intellectual property (IP) and commercial advantage.

Networks and Network Analysis

Because networks are such a common component of SoS, they warrant specific attention. In SoS that are based on an underlying network, communications and information exchange typically constitute a SoS in its own right. This enabling SoS requires architecting like any other SoS, which will be addressed in this section. In the case of an enabling network SoS, the ‘user’, the end-to-end functionality of the larger SoS and enabling network SoS is driven by these user needs. The relationship between SoSE concepts and network enablement, as well as the concepts of networks and network analysis that extend beyond information sharing, have been explored extensively by the defense community (Dickerson and Mavris 2009). For instance, during the U.S. Navy’s work on command, control, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR) as part of a SoS (Owens 1996), the term network included organizational aspects of command and control (C2) structure as well as communications.

Differences in the architecting of an enabling network SoS derive from the fact that these SoS are typically built on commercial technologies and architectures, which are changing rapidly in today’s dynamic technological environment. In addition, these enabling networks are often shared among SoS and hence may further constrain the overall SoS architecture. For example, many SoS (for cost and convenience) expect to operate over the internet, and therefore must consider characteristics of the internet in the expectations for performance and security provided by

use of a shared enabling infrastructure.

Enabling network SoS architecting is particularly well-served by the initial analysis that explores sensitivities through modeling, simulation, analysis, and/or laboratory experimentation and identifies scalability issues or divergent behavior (e.g., concerning requirements or usage assumptions, assumed network bandwidth, or others), beyond which performance starts to break down. This type of analysis provides a basis for network architecture decisions.

In directed SoS, because of the top-down control, there is the option for creating a specialized network for the particular SoS. In the other types of SoS, if the constituents are already supported by some type of a network then the overall SoS networking approach typically needs to accommodate these since the constituent systems are likely to need to continue to use their current approach to support their original users.

Interoperability

Interoperability within a SoS implies that each system can communicate and interact (control) with any other system regardless of their hardware and software characteristics or nature. This implies that each constituent member (and potential new members) of a SoS should be able to communicate with others without compatibility issues in the operating systems, communication hardware, and so on. For this purpose, a SoS needs a common language the SoS's systems can speak. Challenges here are to work towards a common language for exchange of information and data among systems of a SoS. Examples of such system are XML (eXtensible Markup Language), as one potential candidate (Jamshidi, 2009a).

However, interoperability must be achieved at many levels and not just at the data/network level. There are a number of frameworks that describe the levels of interoperability. From military applications, the NCOIC (Network Centric Operations Industry Consortium) Interoperability Framework (NCOIC 2008) covers three broad levels of interoperability, subdivided into further layers as indicated below:

- Network Transport:
  - Physical Interoperability and
  - Connectivity and Network Interoperability;
- Information Services:
  - Data/Object Model Interoperability,
  - Semantic/Information Interoperability, and
  - Knowledge/Awareness of Actions Interoperability; and
- People, Processes and Applications:
  - Aligned Procedures,
  - Aligned Operations,
  - Harmonized Strategy/Doctrine, and
  - Political or Business Objectives.

This spectrum of interoperability layers requires appropriate coherence at each layer consistent with the SoS shared goals.

There exist interoperability frameworks in other fields of activity. An example is the European Interoperability Framework (European Commission 2004), which focuses on enabling business (particularly e-business) interoperability and has four levels within a political context:

- Legal Interoperability,
- Organizational Interoperability,
- Semantic Interoperability, and
- Technical Interoperability.

The interoperability between the component systems of a SoS is a fundamental design consideration for SoS that may be managed through the application of standards.

References

Works Cited

Abel, A., and S. Sukkarieh. 2006. "The Coordination of Multiple Autonomous Systems using Information Theoretic Political Science Voting Models." Proceedings of the IEEE International Conference on System of Systems Engineering, April 24-26, 2006, Los Angeles, CA, USA.

Azani, C. 2009. *A Multi-criteria Decision Model for Migrating Legacy System Architectures into Open Systems and Systems-of-Systems Architectures.* Washington, DC, USA: Defense Acquisition University.

Azarnoush, H., B. Horan, P. Sridhar, A.M. Madni, and M. Jamshidi. 2006. "Towards optimization of a real-world robotic-sensor system of systems". Proceedings of the World Automation Congress (WAC), July 24-26, 2006, Budapest, Hungary.

Cloutier, R.M., J. DiMario, and H.W. Polzer. 2009. "Net-Centricity and System of Systems," in *Systems of Systems Engineering - Principles and Applications*, edited by M. Jamshidi. Boca Raton, FL, USA: CRC Press.

Dagli, C.H., and N. Kilicay-Ergin. 2009. "System of Systems Architecting," in *Systems of Systems Engineering - Principles and Applications*, edited by M. Jamshidi. Boca Raton, FL, USA: CRC Press.

Dickerson, C.E., and D. Mavris. (2009) *Architecture and Principles of Systems Engineering*. New York, NY, USA: CRC Press, Auerbach Publications.

DoD. 1998. *Levels of Information System Interoperability*. Washington, DC, USA: C4IST Interoperability Working Group, US Department of Defense.

Hall, J. 2007. "Openness – An Important Principle For The Stewardship of DoD IT Standards." *DSPO Journal*, 4-7. Available: http://www.dsp.dla.mil/app_uil/content/newsletters/journal/DSPJ-01-07.pdf.

Henshaw, M. (ed.). 2011. *Assessment of open architectures within defence procurement issue 1: systems of systems approach community forum working group 1 - open systems and architectures.* London, UK: SoSA Community Forum Working Group 1 (Crown owned copyright). Available: https://dspace.lboro.ac.uk/2134/8828.

Hitchins, D.K. 2003. *Advanced Systems Thinking, Engineering and Management*. Norwood, MA, USA: Artech House, Inc.

IEEE. 1990. IEEE 610.12-1990, *Standard Glossary of Software Engineering Terminology*. Washington, DC, USA: Institute of Electrical & Electronics Engineers (IEEE).

Jamshidi, M. (ed.) 2009. *Systems of Systems Engineering - Principles and Applications*. Boca Raton, FL, USA: CRC Press.

Johnson M. 2009. "System of Systems Standards," in *System of Systems Engineering - Innovations for the 21st Century*. Hoboken, NJ, USA: Wiley.

Lopez D. 2006. "Lessons Learned From the Front Lines of the Aerospace." Proceedings of the IEEE International Conference on System of Systems Engineering, April 24-26, 2006, Los Angeles, CA, US.

Mittal, S. 2000. "DEVS Unified Process for Integrated Development and Testing of Service Oriented Architectures." PhD Dissertation. Tucson, AZ, USA: University of Arizona.

NCOIC. 2008. "NCOIC Interoperability Framework (NIF(R))." Available: http://www.ncoic.org/technology/technical-products/frameworks/10-technology/33-tech-prod-framework-nif.

Owens, W.A. 1996. *The Emerging U.S. System-of-Systems*. Washington, DC, USA: The National Defense University, Institute of National Security Studies.

Rebovich Jr., G. 2009. "Chapter 6: Enterprise System of Systems," in *Systems of Systems Engineering - Principles and Applications.* Boca Raton, FL, USA: CRC Press. p. 169.

Sahin, F., M. Jamshidi, and P. Sridhar. 2007. "A Discrete Event XML based Simulation Framework for System of Systems Architectures." Proceedings of the IEEE International Conference on System of Systems, April 16-18, 2007, San Antonio, TX, USA.

US Department of Defense. 2008. *Systems Engineering Guide for Systems of Systems,* version 1.0. Washington, DC, USA: US Department of Defense.

Wells, G.D., and A.P. Sage. 2008. "Engineering of a System of Systems," in *Systems of Systems Engineering - Principles and Applications.* Boca Raton, FL, USA: CRC Press.

Wojcik, L.A., and K.C. Hoffman. 2006. "Systems of Systems Engineering in the Enterprise Context: A Unifying Framework for Dynamics." Proceedings of the IEEE International Conference on System of Systems Engineering, April 24-26, 2006, Los Angeles, CA, USA.

Zachmann, J. 1987. "A framework for information systems architecture." *IBM Systems Journal*. 26 (3).

Zeigler, B.P., T.G. Kim, and H. Praehofer. 2000. *Theory of Modeling and Simulation*. New York, NY, USA: Academic Press.

Primary References

Chen, D., G. Doumeingts, F. Vernadat. 2008. "Architectures for Enterprise Integration and Interoperability: Past, Present and Future." *Comput.Ind*. 59 (7):647-659.

Maier, M.W. 1998. "Architecting Principles for Systems-of-Systems." *Systems Engineering.* 1 (4): 267-284.

Additional References

Dickerson, C.E., S.M. Soules, M.R. Sabins, and P.H. Charles. 2004. *Using Architectures for Research, Development, and Acquisition*. Washington, DC, USA: Office of the Assistant Secretary of The Navy (Research Development And Acquisition). ADA427961. Available: http://handle.dtic.mil/100.2/ADA427961.

European Commission. 2010. "Annex to the Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of Region," in *Towards interoperability for European public services.* Available: http://ec.eupora.eu/isa/strategy/doc/annex_ii_eif_en.pdf

Giachetti, R.E. 2010. *Design of Enterprise Systems, Theory, Architecture, and Methods*. Boca Raton, FL, USA: CRC Press.

Rhodes, D.H., A.M. Ross, and D.J. Nightingale. 2010. "Architecting the System of Systems Enterprise: Enabling Constructs and Methods from the Field of Engineering Systems." IEEE International Systems Conference, March 23-26, 2009, Vancouver, Canada.

MITRE. 2012. "Architectures Federation," in *Systems Engineering Guide.* Bedford, MA, USA: MITRE Corporation. Accessed September 11, 2012. Available: http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/engineering_info_intensive_enterprises/architectures_federation.html.

Mittal, S. 2000. "Extending DoDAF to Allow DEVS-Based Modeling and Simulation." *Journal of Defense Modeling and Simulation (JDMS).* 3 (2).

Valerdi R., E. Axelband, T. Baehren, B. Boehm, and D. Dorenbos. 2008. "A research agenda for systems of systems architecting." *International Journal of System of Systems Engineering*. 1 (1-2): 171-188.

Zeigler, B.P., D. Fulton, P. Hammonds, and J. Nutaro. 2000. "Framework for M&S–Based System Development and Testing in a Net-Centric Environment." *ITEA Journal of Test and Evaluation*. 26 (3): 21-34.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Socio-Technical Features of Systems of Systems

Most systems of systems (SoS) are socio-technical systems that are composed of a number of interdependent resources, such as, people, processes, information, and technology that must interact with each other and their environment in support of a common mission (glossary) (See also Enterprise Systems Engineering).

Managing the Socio-Technical Features of Systems of Systems

The managerial and operational independence of constituent systems of a SoS (Maier, 1998) imply that in general, SoS are socio-technical systems. Turning to views outside of Systems Engineering, Ergonomists regard socio-technical systems as having the following characteristics (Maguire, 2014):

- There are collective operational tasks,
- They contain social and technical sub-systems,
- They are open systems (i.e. strongly interacting with their environments), and
- The concept of the system being an unfinished system.

These are also characteristics of Systems of Systems. Klein (2014) has noted that approaches to socio-technical systems can take the two perspectives of “system affects people” or “people affect system”, depending upon how the system boundary is drawn. It is generally true for systems that consideration of their context requires socio-technical aspects to be taken into account.

Although there are many matters concerning the socio-technical aspects of SoS, there are two main issues, which are somewhat related, requiring attention. The first is the need for appropriate governance structures, given that operational and/or managerial independence affects top-down direction of the SoS and may compromise achievement of the SoS goal(s). The second issue is a lack of situational awareness of managers, operators, or other stakeholders of the SoS, so that they may not understand the impact of their local decisions on the wider SoS.

SoS Governance

Generally, design and operation of complex systems is concerned with control, but the classification of SoS (Dahmann, et. al., 2008) is based on the notion of diminishing central control, as the types go from directed to virtual. Sauser, et. al. (2009) has described the ‘control paradox of SoS’ and asserted that for SoS, ‘management’ is replaced by ‘governance’. Thus in C2 terms, the emphasis in SoS is on command, rather than control: ‘Control is a function of rules, time, and bandwidth; whereas command is a function of trusts, influence, fidelity, and agility’. Trusts, influence, and fidelity are intrinsically human qualities. The matters relating to the organizational aspects, such as these, are relevant to Enterprise Systems . Siemieniuch and Sinclair (2014) highlighted the complexity issues with respect to SoS by drawing attention to a variety of organizational characteristics that concern the interactions among system operators in a SoS:

- Many agents, of different kinds
- Some degree of behavioral autonomy for agents
- Multiple steady states for agents
- Interactions between agents in an environment
- Lots of connections between agents
- Agents communicating in parallel
- Effects of an evolving environment

- Effects of evolving agents
- Interactions between different goals within an agent
- Interactions between agents with different goals
- Language/culture differences between agents

A major governance issue for SoS is understanding the ownership of, and making reliable estimates of risk (Fovino & Masera, 2007). High levels of connectivity, and the potential for emergent behavior due to the interactions of separately owned/operated constituent systems, means that significant risks may go unacknowledged and their mitigations unplanned.

In general, governance can be summed up by asking three connected questions (Siemieniuch and Sinclair, 2014):

- Are we doing the right things (leadership)?
- Are we doing those things right (management)?
- How do we know this (metrics and measurements)?

Currently, there is no accepted framework for addressing these questions in a SoS context, but Henshaw et. al. (2013) have highlighted architectures as an important means through which governance may be clarified. They postulate that a SoS can be regarded as a set of trust and contract relationships between systems (i.e. including both informal and formal relationships). The systems architect of a constituent system must, therefore, address trust issues for each participating organization in the overall enterprise with which his/her system must interoperate. For SoS, technical engineering governance is concerned with defining and ensuring compliance with trust at the interface between constituent systems. An example of difficulty managing the interfaces in a SoS is provided in the Cassini-Huygens mission case study .

Situational Awareness

Situational awareness is a decision maker's understanding of the environment in which he/she takes a decision; it concerns information, awareness, perception, and cognition. Endsley (1995) emphasizes that situational awareness is a state of knowledge. There are numerous examples of SoS failure due to the operator of one constituent system making decisions based on inadequate knowledge of the overall SoS (big picture). Fratricide incidents are among the most shocking (see for example Rasmussen, 2007).

On the other hand, SoS development is also viewed as the means through which improved situational awareness may be achieved (Van der Laar, et. al., 2013). In the defense environment, Network Enabled Capability (NEC) was a system of systems approach motivated by the objective of making better use of information sharing to achieve military objectives. Court (2005) identified that the NEC benefits chain was concerned with the quality of decision making, which relies on the quality of shared awareness, which relies, in turn, on networks and information quality; a crucial element is the sharing of information, knowledge, and understanding. NEC is predicated on the ability to share useful information effectively among the stakeholders that need it. It is concluded that improving situational awareness will improve SoS performance, or at least reduce the risk of failures at the SoS level. Thus, the principles which govern the organization of the SoS should support sharing information effectively across the network; in essence, ensuring that every level of the interoperability spectrum is adequately serviced. Operators need insight into the effect that their own local decisions may have on the changing SoS or environment; similarly they need to understand how external changes will affect the systems that they own.

References

Works Cited

Court, G. (2005) Validating the NEC benefits chain in 11th ICCRTS. Cambridge, UK. Retrieved from http://www.dodccrp.org/events/11th_ICCRTS/html/papers/155.pdf

Dahmann, J. S. & Baldwin, K. J. (2008) Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering, 2nd Annual IEEE Systems Conference, 1–7. http://doi.org/10.1109/SYSTEMS.2008.4518994

Endsley, M. R. (1995) Toward a Theory of Situation Awareness in Dynamic Systems, J. Human Factors and Ergonomics Soc., 37(1), 32–64. http://doi.org/10.1518/001872095779049543

Fovino, I. N., & Masera, M. (2007) Emergent disservices in interdependent systems and system-of-systems, in Proc. IEEE International Conference on Systems, Man and Cybernetics, Vol. 1, pp. 590–595. http://doi.org/10.1109/ICSMC.2006.384449

Henshaw, M. J. de C., Siemieniuch, C. E., & Sinclair, M. A. (2013) Technical and Engineering Governance in the Context of Systems of Systems, in NATO SCI Symp. Architecture Assessment for NEC (pp. 1–10). Tallinn, Es. NATO STO.

Henshaw, M. (2014) A Socio-Technical Perspective on SoSE, in Lecture Series in Systems of Systems Engineering for NATO Defence Applications (SCI-276). NATO CSO.

Klein, L. (2014) What do we actually mean by ‘sociotechnical’? On values, boundaries and the problems of language, Appl. Ergon., vol. 45, no. 2 PA, pp. 137–142.

Maguire, M. (2014) Socio-technical systems and interaction design - 21st century relevance, Appl. Ergon., vol. 45, no. 2 PA, pp. 162–170.

Rasmussen, R. E. (2007) The Wrong Target, MSc thesis, Joint Advanced Warfighting School, Norfolk, VA. Retrieved from http://www.dtic.mil/dtic/tr/fulltext/u2/a468785.pdf

Rittel, H.W.J., and M.M. Webber. 1973. "Dilemmas in a General Theory of Planning." Amsterdam, The Netherlands: Elsevier Scientific Publishing Company, Inc. p. 155–169, in Developments in Design Methodology, edited by N. Cross, 1984. Chichester, West Sussex, England, UK: John Wiley & Sons, Ltd. p. 135–144.

Sauser, B., Boardman, J., & Gorod, A. (2009) System of Systems Management, in System of Systems Engineering: Innovations for the 21st Century, M. Jamshidi (Ed.), (pp. 191–217) Wiley.

Siemieniuch, C.E. & Sinclair, M.A. (2014) Extending systems ergonomics thinking to accommodate the socio-technical issues of Systems of Systems, Appl. Ergon., V 45, Issue 1, Pages 85-98

Van der Laar, P., Tretmans, J., & Borth, M. (2013) Situational Awareness with Systems of Systems. Springer.

Primary References

Checkland, P.B. 1981. Systems Thinking, Systems Practice. Chichester, West Sussex, England, UK: John Wiley & Sons, Ltd.

Hubbard, E-M., C.E. Siemieniuch, M.A. Sinclair, and A. Hodgson. 2010. "Working towards a Holistic Organisational Systems Model." Presented at 5th Int. Conf. Systems of Systems Engineering (SoSE), 22-24 June, 2010, Loughborough, UK.

Rittel, H.W.J., and Webber, M.M. 1973. "Dilemmas in a General Theory of Planning," in *Policy Sciences 4*. Amsterdam, The Netherlands: Elsevier Scientific Publishing Company, Inc. p. 155–169. In Cross, N. 1984. Ed. *Developments in Design Methodology*. Chichester, West Sussex, England, UK: John Wiley & Sons, Ltd. p. 135–144.

Additional References

Bruesburg, A., and G. Fletcher. 2009. *The Human View Handbook for MODAF*, draft version 2, second issue. Bristol, England, UK: Systems Engineering & Assessment Ltd. Available: http:/ / www. hfidtc. com/ research/ process/reports/phase-2/hv-handbook-issue2-draft.pdf.

IFIP-IFAC Task Force. 1999. "The Generalised Enterprise Reference Architecture and Methodology," V1.6.3. Available: http://www.cit.gu.edu.au/~bernus/taskforce/geram/versions/geram1-6-3/v1.6.3.html.

ISO. 1998. ISO 14258:1998, *Industrial automation systems — Concepts and rules for enterprise models.* Geneva, Switzerland: International Organization for Standardization.

ISO. 2006. ISO 19439:2006, *Enterprise integration — Framework for enterprise modelling.* Geneva, Switzerland: International Organization for Standardization.

ISO. 2007. ISO 19440:2007, *Enterprise integration — Constructs for enterprise modelling.* Geneva, Switzerland: International Organization for Standardization.

Miller, F.P., A.F. Vandome, and J. McBrewster. 2009. *Enterprise Modelling.* Mauritius: Alphascript Publishing, VDM Verlag Dr. Müller GmbH & Co. KG.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Capability Engineering

Capability Engineering Perspectives

The term capability is widely used across many industrial sectors and has begun to take on various specific meanings across, and even within, those sectors. Terms such as capability-based acquisition, capability engineering and management, life capability management, capability sponsor, etc. are now ubiquitous in defense and elsewhere. Henshaw et al. (2011) have identified at least eight worldviews of capability and capability engineering and concluded that the task of capability engineering is not consistently defined across the different communities.

Whilst most practitioners recognize that there is a strong relationship between capability and system of systems (SoS), there is no agreed position; however, there are two beliefs that are widely accepted among the different communities, including:

- a capability comprises a range of systems, processes, people, information and organizations. (i.e. a system at levels three through five in Hitchin's (2003) five layer model, such as a Carrier-Strike capability) and
- the capability is an emergent property of SoS (i.e. the capability of Carrier-Strike to engage targets within 300 miles of the sea.)

Services View of SoSE

As it has been discussed throughout the Systems of Systems (SoS) knowledge area, a 'system of systems' is typically approached from the viewpoint of bringing together multiple systems to provide broader capability. As is discussed in Architecting Approaches for Systems of Systems, the networking of the constituent systems in a SoS is often a key part of an SoS. In some circumstances, the entire content of a SoS is information and the SoS brings together multiple information systems to support the information needs of a broader community. These 'information technology (IT)-based' SoSs have the same set of characteristics of other SoSs and face many of the same challenges. Currently, IT has adopted a 'services' view of this type of SoS and increasingly applies a International Organization for Standaradization (ISO) 20000 series (Information technology -- Service management) or

Information Technology Infrastructure Library (ITIL) v. 3 (OGC 2009) based approach to the design and management of information-based SoS. A service perspective simplifies SoSE as it:

- is a more natural way for users to interact with and understand a SoS,
- allows designers to design specific services to meet defined performance and effectiveness targets, and
- enables specific service levels to be tested and monitored through life.

Although it has not been proven to be universally applicable, the services view works well in both IT and transportation SoS.

References

Works Cited

Erl, T. 2008. *SOA Principles of Service Design.* Boston, MA, USA: Prentice Hall Pearson Education.

Hitchins, D.K. 2003. *Advanced Systems Thinking, Engineering and Management*. Norwood, MA, USA: Artech House, Inc.

OGC (Office of Government Commerce). 2009. *ITIL Lifecycle Publication Suite Books.* London, UK: The Stationery Office.

Primary References

Henshaw, M., D. Kemp, P. Lister, A. Daw, A. Harding, A. Farncombe, and M. Touchin. 2011. "Capability Engineering - An Analysis of Perspectives." Presented at International Council on Systems Engineering (INCOSE) 21st International Symposium, June 20-23, 2011, Denver, CO, USA.

Additional References

Davies, J.K. 2011. *Survey of Background Literature for Capability Engineering*. INCOSE UK Capability Working Group Report.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Healthcare Systems Engineering

Healthcare Systems Engineering

This article provides an overview of the role of systems engineering in the engineering or re-engineering of healthcare systems to meet a number of modern day challenges. The role of SE in medical devices, healthcare IT, pharmaceuticals, and public health systems are considered and contrasted to "traditional" SE practices discussed elsewhere in the SEBoK. See Overview of the Healthcare Sector for details of the stakeholders and constraints of the these different parts of the sector.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Overview of the Healthcare Sector
- Systems Engineering in Healthcare Delivery
- Systems Biology
- Lean in Healthcare

Healthcare and Systems Engineering

Healthcare today faces many challenges related to safety (e.g. Hospital Safety Score 2013, Andel et al. 2012, Institute of Medicine 1999), affordability, access, and the means for reliably producing positive outcomes for all patients of all ages and across all care environments. Furthermore, the health of individuals is challenged by many threats such as environmental and behavioral norms, emerging natural infectious diseases, and acute and chronic conditions that are becoming more prevalent because of longer lifespans. Re-engineering today’s healthcare to address these challenges requires a systems approach – an approach that develops solutions to contend with the complexity of healthcare-related policy, economics, social dynamics, and technology. Systems Engineers are trained to grapple with this kind of complexity by thinking holistically and to work with trans-disciplinary teams to develop solutions making re-engineering healthcare a natural fit for systems engineers and the tools of systems engineering.

The disciplines involved in re-engineering healthcare are far reaching across academia, government, industry, private, and public sectors including the patients and families the healthcare field serves. Systems Engineers involved in this re-engineering draw on several tools when working with these stakeholders to develop solutions. In doing so they follow the general systems principles described in the Systems Approach Applied to Engineered Systems knowledge area in SEBoK Part 2. First, with so many diverse stakeholders involved in this field, it is vitally important for the Systems Engineer to help clarify the problem or opportunity and to conceive of the objective of the re-engineering. They need to “envision the solution” without being entirely prescriptive of the solution’s specific implementation, see Identifying and Understanding Problems and Opportunities. Then, drawing from best practices, the Systems Engineer guides the stakeholders through the synthesis of possible solutions and the analysis and selection between alternatives. Systems engineers are also involved in the implementation and testing and the deployment, use and sustainment of healthcare systems to provide stakeholder value. The systems approach in healthcare must be particularly mindful to not exclusively focused on technical aspects of the effort since the solutions to healthcare’s challenges exist not only in technical areas but the integration of culture, workflow and

processes, with technology as a tool to support the delivery of safe, affordable, and accessible care.

To achieve this system approach healthcare projects follow a version of the SE life cycle described in SEBoK Part 3. This included the creation of Stakeholder and System Requirements, Systems Architecture and Design and System Integration, Verification and Validation. The SE life cycle extends to include System Deployment, Operation, Maintenance and Logistics. Healthcare project will also follow some of the Systems Engineering Management processes described in Part 3.

It is vitally important for the healthcare systems engineer to ensure socio-technical integration and interoperability among system components are part of any project – the last thing healthcare needs is another standalone innovation that perpetuates the silos that exist in the field today. Remaining focused on the objective and problem to solve, managing scope creep, disciplined design, implementation, and project management are key activities the Systems Engineer is responsible for in healthcare systems engineering.

Systems Engineering for Medical Device Development

Systems Engineering for medical device development is essentially an application of Product Systems Engineering as described in SEBoK Part 4 with a few customizations:

- The life cycle has to comply with specific healthcare regulations, which constrain aspects of the life cycle, as exemplified by FDA regulations in the US (21CFR 820.30)
- The products are market driven, with little customization allowed by the manufacturer at the customer site
- The markets are midsized, with the market for a given technology or product line often being in the $1-10B range
- Medical device development programs are mid-sized…many from 10-100 man-years of development, lasting 1-2 years
- Time to market is critical, with the first mover or first with a complete solution capturing the majority of the profits
- Most products are cyber-physical, with software becoming a larger part of the product. Many products include significant aspects of physiology or chemistry
- There is a special tension between "efficacy" and "safety". Efficacy requires the vast majority to be helped. Safety is compromised if only a very small minority is adversely affected. Truly safe systems require a special approach to systems engineering . (Leveson 2011)
- Customer feedback may be constrained by safety issues as well as HIPAA regulations

Device Development in a Market Environment

One critical difference between many "traditional" systems engineering industries (defense and aerospace) and healthcare device development is that most healthcare device development is market driven, rather than contract driven. Some key differences between market and contract systems engineering:

- The program size (budget) and dates are not 'fixed', they are set by the business leadership designed to maximize return on investment across a portfolio of product programs
- Program scope and requirements are not fixed externally; they can be changed fairly rapidly by negotiation between functions and the executive committee.
- The goal for the product development isn't necessarily a feature set, it is a market share and price premium relative to the competition…which can be a moving target. A competitive announcement will often force a change in the program scope
- In a contract based program there is an identified customer, with a set of applications and workflows. In a market driven program the workflow and use cases are defined by the developer, and the buyer needs to 'own' the integration of the offering into their specific systems and workflows.
- For specific medical products the FDA can require pre-market trials and post market studies . (FDA 2014)

- The different types of healthcare reimbursement across the world (universal coverage private insurance, national single provider, national single payer, private insurance, and out of pocket) creates dramatically different market dynamics (for individuals, healthcare providers, and product developers) . (Reid 2010)

Regulations for Medical Device Development

As with all regulated products, there are many regulations governing the development of medical devices. The medical device industry specific regulations are primarily driven by the US (FDA), Europe (European Commission), and Canada (Health Canada). Within the US, the FDA governs medical devices primarily through 21 CFR 820.30 (Quality Systems Regulation, Subpart C Design Controls) , which contains requirements similar to ISO 13485. The sections of the Quality Systems Regulation for Design Controls can be mapped fairly directly to ISO/IEC/IEEE 15288 (2015) and the INCOSE *SE Handbook* (INCOSE 2015).

Table 1. Comparison of Healthcare Safety Regulations with ISO/IEC/IEEE 15288 and the INCOSE *SE Handbook*.

21CFR820.30	ISO/IEC/IEEE 15288:2015	INCOSE *SE Handbook*
(b) Design and development planning	6.3.1 Project Planning Process	5.1 Project Planning Process
(c) Design input.	6.4.2 Stakeholder needs and requirements definition process 6.4.3 Systems requirements definition process	4.2 Stakeholder needs and requirements definition process 4.3 Systems requirements definition process
(d) Design output	6.4.5 Design definition process 6.4.7 Implementation process	4.5 Design definition process 4.7 Implementation process
(e) Design review	6.3.2 Project Assessment and Control process	5.2 Project Assessment and Control process
(f) Design verification	6.4.9 Verification Process	4.9 Verification Process
(g) Design validation	6.4.11 Validation Process	4.11 Validation Process
(h) Design transfer	6.4.10 Transition Process	4.10 Transition Process
(i) Design changes	6.3.5 Configuration Management Process 6.4.13 Maintenance Process	5.5 Configuration Management Process 4.13 Maintenance Process
(j) Design history file	6.2.6 Knowledge Management Process	5.6 Information Management Process

In the biomedical and healthcare environment, an important differentiator in Risk Management activities compared to other industries (see Risk Management) is that the users and patients are the center of risk analysis rather than technical or business risks. Risk management is an important element of the design control process, as preliminary hazard analysis drive initial design inputs. Traceability between identified risks, risk mitigations, design inputs, and design outputs is a key factor in product clearance through regulatory agencies. Most regulatory bodies have recognized ISO 14971: Medical devices -- Application of risk management to medical devices as a methodology for assessing and documenting product safety and effectiveness.

Usability Engineering is an important subset of risk management activities. ISO 62366-1 Medical devices – Part 1: Application of usability engineering to medical devices provides a “process for a manufacturer to analyze, specify, develop and evaluate the usability of a medical device as it relates to safety. This usability engineering (human factors engineering) process permits the manufacturer to assess and mitigate risks associated with correct use and use errors, i.e., normal use.“ . (IEC 62366-2015) For example, for a device designed for home care use, there are many complex interfaces that product designers must consider. Patients may be physically or cognitively affected (age, medication, injury, etc.); they may be untrained or cared for people who are untrained; they are not professionals used to technical systems, etc. Even in the hospital setting, untrained patients may have physical access to systems. This puts a critical focus on usability and human factors considerations and the complexity of the use environment.

Further, as medical devices incorporate more software and become cyber-physical devices, the regulators are also focusing on privacy and security (ISO 21827) and software life cycle management (ISO 62304).

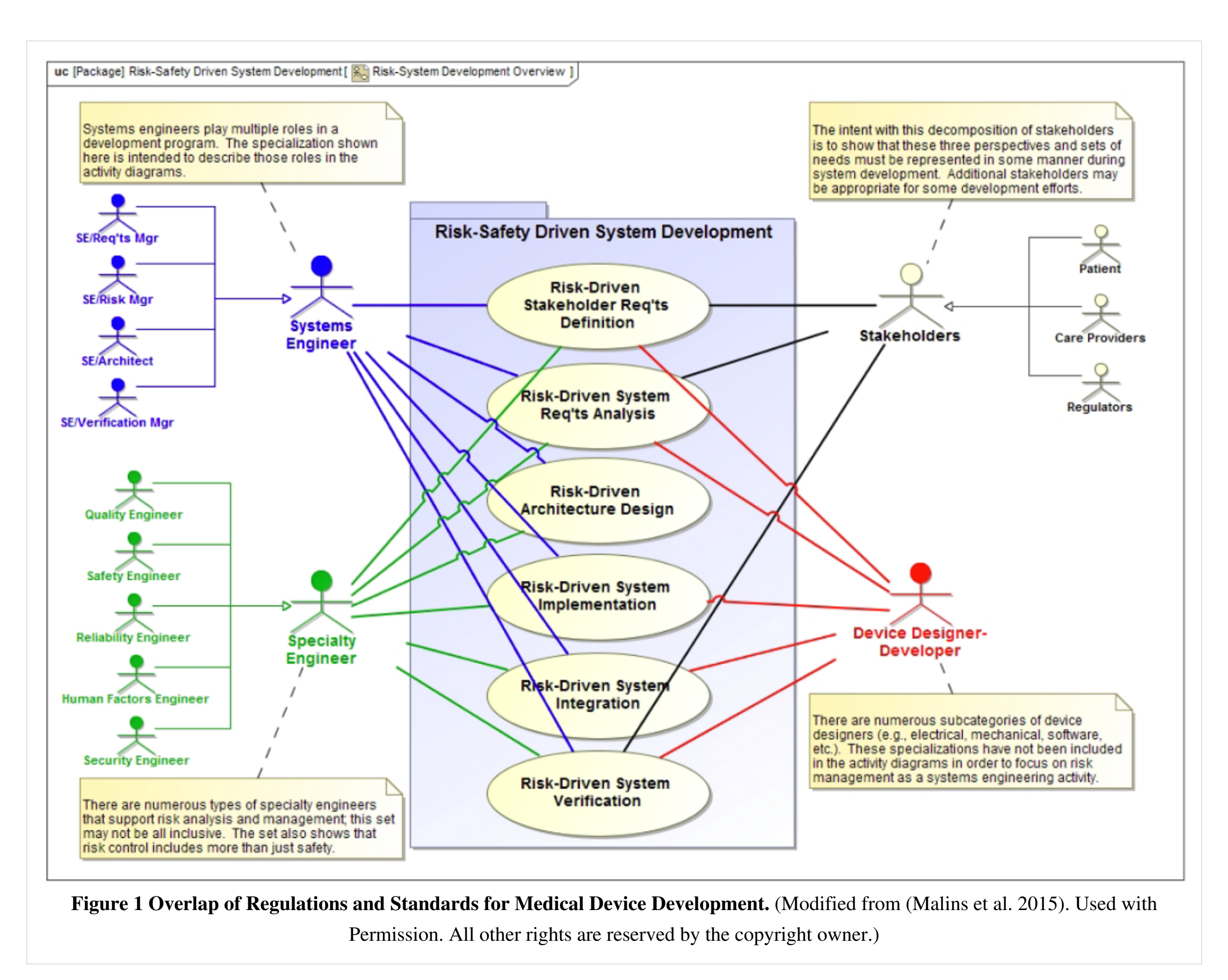


Figure 1 Overlap of Regulations and Standards for Medical Device Development. (Modified from (Malins et al. 2015). Used with Permission. All other rights are reserved by the copyright owner.)

Medical Device regulations, guidance, and technical standards are constantly changing, adding a complex dynamic to manage and incorporate throughout the product development life cycle.

Systems Engineering for Healthcare IT

Systems Engineering for Healthcare Information Technology is very similar to other IT developments, with the addition of medical regulations. Healthcare Information Technology is critical to efficient flow of information and delivery of services . (Presidents Council of Advisors on Science and Technology 2010) The product development is a mix of contract driven development (with a target customer, such as healthcare.gov), and market driven (where there are more standard products, with minimal customization). Much of the market, especially for hospitals and hospital chains, is a mix of standard products with large amounts of customization to the customer's specific needs, terminology, and workflows.

Systems Engineering for Pharmaceuticals

The pharmaceutical industry leverages systems that include hardware, software and sometimes single-use components in different part of their value chain, for example complex analytical systems during drug discovery, complex bioreactors and downstream filtration and chromatography systems in manufacturing and drug delivery devices for the use of their drugs. These systems are subject to very different regulations, e.g. GMP or medical devices, depending on the use. One challenging aspect of these systems is that the users have different skill sets and working under different environments. And in all of the examples below, biological and/or chemical processes run on these systems, requiring deep domain knowledge of the system development teams.

The in-vitro diagnostic industry also uses many systems, small devices (e.g. self-testing blood glucose or coagulation monitoring systems) all the way to large, fully automated, high throughput systems for the use in centralized laboratories. Very often, these systems operate as a closed system, so that the reagents used for the diagnostics tests, are proprietary and the vendor of the system only guarantees high quality results only when using the proprietary chemistry. This enables the vendors to often 'place' the instruments as highly competitive prices when the actual profit is generated through the consumables.

For the chemistry part of pharma, understanding the scientific method, using a systems thinking approach, and using six sigma approaches to managing variation and interdependencies is critical. Once you create a product which includes software and physical parts (including manufacturing equipment), systems engineering of the functional design, design analysis, and integration and verification of the solution become critical.

Systems Challenges for Public Health

Summits and inquiries into problems or shortcomings in the public health space have consistently uncovered the same issues: systemic failures in the way that public health is approached that make it nearly impossible to adequately respond to major health events. Examples can be seen from the US response to Hurricane Katrina (e.g. The White House 2006), the 2011 Thoku tsunami (e.g. Carafano 2011, The Heritage Foundation 2012), or even the 2014-2015 Ebola outbreak in West Africa (e.g. GHTC 2015).

The White House report provides insights into just a few of potential challenges for the health aspects of disasters or large-scale emergencies (2006, Chapter 6):

- Tens of thousands of people may require medical care.
- Large portions of a population with chronic medical conditions may themselves without access to their usual medications and sources of medical care.
- Hospitals and other healthcare facilities may be totally destroyed or otherwise rendered inoperable and the area's health care infrastructure may sustain extraordinary damage.

The types of public health challenges will also change over time: Immediate challenges include the identification, triage and treatment of acutely sick and injured patients; the management of chronic medical conditions in large numbers of evacuees with special health care needs; the assessment, communication and mitigation of public health risk; and the provision of assistance to State and local health officials to quickly reestablish health care delivery systems and public health infrastructures. (The White House 2006) As time passes, longer-term infectious disease outbreaks may occur or environmental impacts may cause health risks (e.g. Fukushima nuclear meltdown after the 2010 tsunami). And over time, the public health and overall healthcare infrastructure must be re-established and repaired.

But the public health "system" in most countries, as currently structured, is not prepared to deal with these types of challenges. In talking about the US, Salinsky and Gursky state, "Despite recent attention to the biodefense role of public health, policymakers have not developed a clear, realistic vision for the structure and functionality of the governmental public health system. Lack of leadership and organizational disconnects across levels of government have prevented strategic alignment of resources and undermined momentum for meaningful change. A transformed public health system is needed to address the demands of emergency preparedness and health protection. … The future public health system cannot afford to be dictated by outmoded tools, unworkable structures, and outdated staffing models." (2006)

The framing of the challenge as a systems one requires the application of a systems approach, and the use of tools capable of supporting systems views, to enable better understanding of the challenges for public health and for creating ways to address these challenges. The SEBoK knowledge areas on Enterprise Systems Engineering and Systems of Systems (SoS) at least partially consider some of these challenges from a systems engineering perspective.

Systems Biology for Healthcare

As systems science is a foundation for system engineering, systems biology is becoming recognized as a foundational discipline for healthcare systems engineering. Systems biology is an emerging discipline and is recognized as strategically important when tackling complex healthcare problems. The development of systems biology is also an emerging environment for systems engineers.

According to Harvard University, “Systems biology is the study of systems of biological components, which may be molecules, cells, organisms or entire species. Living systems are dynamic and complex and their behavior may be hard to predict from the properties of individual parts. To study them, we use quantitative measurements of the behavior of groups of interacting components, systematic measurement technologies such as genomics, bioinformatics and proteomics, and mathematical and computational models to describe and predict dynamical behavior. Systems problems are emerging as central to all areas of biology and medicine.” (Harvard University 2010)

As systems biology matures, its integration into healthcare approaches is expected to lead to advanced practices such as personalized and connected healthcare and the resolution of complex diseases.

Conclusion

While systems engineering practices apply to the healthcare domain, they face different challenges than other industries and need to be tailored. In fact, different segments of the healthcare industry can take significantly different approaches to effective systems engineering and systems thinking.

References

Works Cited

21 CFR 820.30. “Part 820 – Quality System Regulation: Subpart C – Design Controls.” *Title 21 – Food and Drugs: Chapter I – Food and Drug Administration, Department of Health and Human Services: Subchapter H – Medical Devices*. Available at: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfCFR/CFRSearch.cfm?fr=820.30

Andel, C., S.L. Davidow, M. Hollander, D.A. Moreno. 2012. “The economics of health care quality and medical errors.” *Journal of Health Care Finance*. 39(1):39:50. Abstract available at: http://www.ncbi.nlm.nih.gov/pubmed/23155743 200,000 Americans die from preventable medical errors including facility-acquired conditions and millions may experience errors. In 2008, medical errors cost the United States $19.5 billion.

Carafono, J.J. 2011. *The Great Eastern Japan Earthquake: Assessing Disaster Response and Lessons for the U.S.* Washington, DC: The Heritage Foundation. Special Report #94 for Japan. May 25, 2011.

FDA. 2014. “Premarket Approval (PMA)”. Washington, DC: U.S. Food and Drug Administration (FDA). Accessed February 17, 2016. Available at: http://www.fda.gov/Medicaldevices/Deviceregulationandguidance/Howtomarketyourdevice/Premarketsubmissions/Premarketapprovalpma/Default.Htm

GHTC. 2015. “Will We Learn from the Lessons of the Ebola Outbreak?” 2015 Policy Report. Washington, DC: Global Health Technologies Coalition (GHTC).

Gursky, E. 2005. *Epidemic Proportions: Building National Public Health Capabilities to Meeting National Security Threats*. Arlington, VA: Analytic Services Inc. (ANSER).

Harvard University. 2010. “Department of Systems Biology.” Cambridge, MA: Harvard University. Accessed February 17, 2016. Available at: https://sysbio.med.harvard.edu/

Hospital Safety Score. 2013. “Hospital Errors are the Third Leading Cause of Death in U.S., and New Hospital Safety Scores Show Improvements are Too Slow.” Washington, DC: The LeapFrog Group. Accessed February 17, 2016. Available at: http://www.hospitalsafetyscore.org/newsroom/display/hospitalerrors-thirdleading-causeofdeathinus-improvementstooslow

IEC. 2015. *Medical devices – Part 1: Application of usability engineering to medical devices*. Geneva, Switzerland: International Electrotechnical Commissions. IEC 62366-1:2015. Available at: http:/ / www. iso. org/ iso/ catalogue_detail.htm?csnumber=63179

INCOSE. 2015. "Section 8.2.2." *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Institute of Medicine. 1999. *To Err is Human: Building a Safe Health System*. Washington, DC: The National Academy Press, The National Academy of Sciences. Novermber 1999. Available at: https:/ / iom. nationalacademies. org/ ~/ media/ Files/ Report%20Files/ 1999/ To-Err-is-Human/ To%20Err%20is%20Human%201999%20%20report%20brief.pdf

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Leveson, N.G. 2011. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: Massachusetts Institute of Technology (MIT).

Presidential Council of Advisors on Science and Technology. 2010. *Report to the President: Realizing the Full Potential of Health Information Technology to Improve Healthcare for Americans: The Path Forward*. Washington, DC: Presidential Council of Advisors on Science and Technology, The White House. December 2010. Available at: https://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-health-it-report.pdf.

Reid, T.R. 2010. *The Healing of America: A Global Quest for Better, Cheaper, and Fairer Health Care*. New York, NY: Penguin Books.

Salinsky, E. and E. Gursky. 2006. "The Case for Transforming Governmental Public Health." *Health Affairs*. 25(4). 1017-2018. Available at: http://content.healthaffairs.org/content/25/4/1017.full

The Heritage Foundation. 2012. *One Year Later: Lessons from Recover After the Great Eastern Japan Earthquake*. Washington, DC: The Heritage Foundation. Special Report #108 on Asia and the Pacific. April 26, 2012.

The White House. 2006. *The Federal Response to Hurricane Katrina Lessons Learned*. Washington, DC: The White House. February 2006. Available at http://library.stmarytx.edu/acadlib/edocs/katrinawh.pdf

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Overview of the Healthcare Sector

This article describes some of the stakeholders of the healthcare sector and the factors which influence the application of systems engineering within it. For an overview of Healthcare Systems Engineering (glossary) and how it deals with these influences see the Healthcare Systems Engineering article.

The healthcare sector is a complex system made up of people, facilities, laws and regulations. It addresses current health, tries to ensure wellness, treats medical problems; creates new medication and medical devices; manages the health both individuals and populations; and helps determine regulations for safety, privacy, the environment, and healthcare delivery itself.

Stakeholders

There are many types of stakeholders in the healthcare sector. The space covers everyone from the general public – who have a stake in their own health and the health of those around them for issues like infectious disease – to the individual researchers who investigate current healthcare problems. The high-level groups of stakeholders include:

- The general public;
- Healthcare providers (such as doctors, nurses, clinics, and hospitals);
- Payers (such as insurance companies);
- Public health organizations;
- Researchers, scientists, and corporations in the pharmaceutical industry;
- Medical device manufacturers;
- Policy makers (particularly those with interest in public health, healthcare safety or privacy policies);
- Healthcare information technology technicians and organizations; and
- Professional organizations and societies relevant to the various aspects of the space.

The healthcare sector is an enormous area financially as well. For example, out of $2.87 trillion on healthcare spent in the US in 2010, the breakdown of components is:

US Healthcare Expenditures in 2010 (information from Emmanual 2014)

Hospital Care	$921B
Physician Services	$555B
Prescription Drugs	$280B
Nursing Home Care	$151B
Other Medical Products	$113B
Dental Services	$93B
Government Public Health	$84B
Other Professional Services	$79B
Home Health Care	$77B
Research	$48B

The sections below provide insight into the landscape for these the stakeholder groups where there is sufficient information currently available. More detail will be added as the healthcare aspects of the SEBoK mature and the team will take particular care to incorporate additional information from outside the US going forward.

Healthcare Delivery

The largest share of the money spent on healthcare in the US healthcare is in hospitals (almost a third). The number of hospitals has been relatively flat for the last 20 years. However, due to the growing cost pressures and increasing paperwork, there has been a general consolidation of hospitals into chains, and independent physician providers into hospitals or group practices. (Emmanuel 2014)

Overall Hospital Landscape (information from (AHA 2014))

Total Number of All U.S. Registered * Hospitals	5,627
Total Number of U.S. Community ** Hospitals	4,926
Total Number of Nongovernment Not-for-Profit Community Hospitals	2,870
Total Number of Investor-Owned (For-Profit) Community Hospitals	1,053
Total Number of State and Local Government Community Hospitals	1,003
Total Number of Federal Government Hospitals	213
Total Number of Nonfederal Psychiatric Hospitals	403

Hospitals range from small community hospitals to the New York-Presbyterian Hospital/Weill Cornell Medical Center with 2,259 beds (Becker 201)], or the University of Pittsburgh Medical Center Presbyterian with $12B in revenue in 2013. (Becker 2013).

Hospital chains tend to be less than 10 hospitals, with less than 10 chains having more than 10 hospitals (Becker 2015). The largest two have almost 200 hospitals (Community Health Systems with 188 and Hospital Corporation of America with 166). The largest systems by revenue are Kaiser Permanente and the Veterans Health Administration with revenue or budget of slightly over $50B each.

Medical Devices Manufacturers

The medical device development landscape is diverse, composed of many markets of intermediate size (many being above $10B in size, with high single digit to double digit growth rates). Some examples are, with projected market sizes in 2020, are:

Types of Medical Devices and Projected Market Share (Emmanuel 2014)

Medical Device Type	***Projected Market Share***
In-vitro diagnostics (IVD)	$75B
Endoscopy	$33B
Interventional Cardiology	$27B
Infection control	$17B
Minimally invasive surgery	$14B
Defibrillators	$13B
Dental Implants	$10B
Infusion pump	$ 7B
Magnetic Resonance	$ 7B
Digital Xray	$ 5B

As described in Healthcare Systems Engineering, this is the area of the healthcare sector that is most closely aligned with classic product-focused businesses.

Healthcare IT

There is a large uncertainty in what constitutes Healthcare IT. The most visible segment is the Electronic Medical Record (EMR) or Electronic Health Record (EHR), but there is also large markets in billing management, clinical decision support, image management, etc. But there is a divergence of market sizes with estimates around $60B [Bain, FierceIT] and some around $104B [Markets and Markets, MedGadget, and PRNewswire].

An EHR installation at a hospital is similar to an Oracle database installation at a company, where much of the cost is customizing the database and workflows to the institution's policies and workflows, and in training the users to the new system and standardized practices which come with IT and automation.

The top 10 Healthcare IT solution providers in 2015 (information from (Healthcare Informatic 2015))

Company	*2015 Revenue*
Optum	$5.2B
Cerner Corp.	$3.4B
McKesson	$3.1B
Dell	$2.9B
Cognizant	$2.7B
Philips	$2.7B
Xerox	$2.4B
Siemens	$2.0B
Epic Systems Corp.	$1.8B
GE Healthcare	$1.5B

Public Health Systems

The World Health Organization (WHO) defines public health as "all organized measures … to prevent disease, promote health, and prolong life among the population as a whole. Its activities aim to provide conditions in which people can be healthy and focus on entire populations, not on individual patients or diseases. Thus, public health is concerned with the total system and not only the eradication of a particular disease." (WHO 2016) Governments at each level define exactly what "public health" will encompass, but typically there are three areas: epidemiology, provision of health services, and workplace and environmental safety and policy. Epidemiology is the study and control health-related events, including disease. Various methods can be used to carry out epidemiological investigations: surveillance and descriptive studies can be used to study distribution; analytical studies are used to study determinants." (WHO 2016, "Health topics: Epidemiology"). Health services may include services such as preventive vaccinations, disease screening, or well-baby or well-child programs. Environmental safety can include developing policies for automobile or workplace safety, monitoring the quality of drinking water, or even conducting restaurant health inspections. In addition to these wide varieties of work, public health organizations are increasingly expected to be responsible for the health-related aspects of disaster and emergency response efforts.

In the US, the public health "system" is really a patchwork of independent healthcare departments. Each state or territory defines the scope and responsibilities of its own public health "department", requiring information and cooperation from hospitals, private physicians, emergency personnel, laboratory networks, and sometimes public

health organizations from other states. (Gursky 2005)

Conclusion

In addition to each group of stakeholders being complex in itself, these stakeholders then interact and work together - or sometimes contradict one another. This makes the landscape of the healthcare systems engineering space itself complex and highlights the need for systems thinking and systems approaches when attempting to address any health-related issues or challenges.

References

Works Cited

AHA. 2014. "Fast Facts on US Hospitals." Chicago, IL: American Hospital Association (AHA). September 2014. Available at: http://www.aha.org/research/rc/stat-studies/fast-facts.shtml

Becker's Healthcare. 2015. "10 largest for-profit hospital systems | 2015". *Becker's Hospital Review*. June 30, 2015. Available at: http://www.beckershospitalreview.com/lists/10-largest-for-profit-hospital-systems-2015.html

Becker's Healthcare. 2014. "100 Largest Hospitals in the US". *Becker's Hospital Review*. August 7, 2014. Available at: http://www.beckershospitalreview.com/lists/8-7-14-100-largest-hospitals-in-america.html

Becker's Healthcare. 2013. "100 Top-Grossing Hospitals in the US". *Becker's Hospital Review*. June 24, 2013. Available at: http://www.beckershospitalreview.com/lists/100-top-grossing-hospitals-in-america-2013.html

Eliades, G., M. Retterath, N. Hueltenschmidt, and K. Singh. 2012. *Healthcare 2020*. Amsterdam, The Netherlands: Bain & Company. Available at: http://www.bain.com/Images/BAIN_BRIEF_Healthcare_2020.pdf.

Emmanuel, E.J. 2014 *Reinventing American Health Care: How the Affordable Care Act will Improve our Terribly Complex, Blatantly Unjust, Outrageously Expensive, Grossly Inefficient, Error Prone System.* New York City, NY: PublicAffairs.

Gold, A. 2014. "Global healthcare IT market projected to hit $66 billion by 2020." FierceHealthIT. April 1, 2014. Available at; http:/ / www. fiercehealthit. com/ story/ global-healthcare-it-market-projected-hit-66-billion-2020/ 2014-04-01.

Gursky, E. 2005. *Epidemic Proportions: Building National Public Health Capabilities to Meeting National Security Threats.* Arlington, VA: Analytic Services Inc. (ANSER). Healthcare Informatics. 2015."2015 HCI 100." Available at: http://www.healthcare-informatics.com/hci100/2015-hci-100-list

Markets and Markets. 2015. *North American Healthcare IT Market by Product (EHR, RIS, PACS, VNA, CPOE, mHealth, Telehealth, Healthcare analytics, Supply Chain Management, Revenue Cycle Management, CRM, Claims Management) by End User (Provider, Payer) - Forecast to 2020.* October 2015. Available at: http:/ / www. marketsandmarkets.com/Market-Reports/north-america-healthcare-it-market-1190.html

MedGadget. 2015. "Global Healthcare IT Market 2020 - Industry Survey, Market Size, Competitive Trends: Radiant Insights, Inc". November 2, 2015. Available at: http:/ / www. medgadget. com/ 2015/ 11/ global-healthcare-it-market-2020-industry-survey-market-size-competitive-trends-radiant-insights-inc.html

PRNewswire. 2015. "Healthcare IT Market Size to Reach $104.5 Billion by 2020: Grand View Research, Inc." October 15, 2015. Available at: http:/ / www. prnewswire. com/ news-releases/ healthcare-it-market-size-to-reach-1045-billion-by-2020-grand-view-research-inc-533012831.html

WHO. 2016. "Health Topics: Epidemiology." Geneva, Switzerland: World Health Organization. Available at: http:// www.who.int/topics/epidemiology/en/

WHO. 2016. "Trade, foreign policy, diplomacy, and health: Public Health." Geneva, Switzerland: World Health Organization. Available at: http://www.who.int/trade/glossary/story076/en/

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Systems Engineering in Healthcare Delivery

The healthcare system is complex and adaptive and confronts significant challenges for which systems engineering tools are useful and necessary. The President's Council of Advisors on Science and Technology (PCAST) prepared a report concluding that healthcare improvement could be accelerated with the use of systems engineering. (PCAST 2014) They noted that they key incentives are wrong (fee for service vs. fee for outcomes), and key enablers are missing (access to useful data, lack accepted systems techniques and people trained in systems engineering)

This article provides an overview of healthcare delivery with some historical context, and describes some different approaches to systems engineering which have been found helpful in addressing healthcare delivery problems.

Human Centered Design

Healthcare delivery is not a Product (glossary) but a Service (glossary) and that makes it different than typical hardware or software design that may be seen in aerospace, defense, or even medical devices. There are three primary factors for these differences. First, quality in services can be difficult to measure objectively. Second, in this service system, care providers are continually making risk, cost, and quality of care decisions at the time of service. Each patient is unique and multiple pathologies and value streams are possible based upon any given patient's needs. Those needs are complemented by a care team that is unique and complex and includes the patients themselves, family support, medical professionals, hospitals, and even the industry in which it resides. Third, if returning to or maintaining wellness is considered to be the core value for a healthcare delivery system, then the patient's behaviors both within and outside any designed care plan has a significant role to play, because roughly half of all healthcare cost is derived from preventable disease. (Conover 2012)

Structure of the Healthcare Delivery Industry

The healthcare industry is large, diverse, and fragmented and this causes considerable complexity. This complexity is experienced both at the macro and micro levels.

At the macro level the healthcare industry is highly fragmented with over 50% of all healthcare workers employed in companies with less than 500 employees. (Griffith & White 2007) Nearly 1 million physicians practice medicine in the US; roughly half of these are in primary care and the rest are in over 30 specialties and many more subspecialties and clinics. In addition to physicians, some 5 million others in some 50 other specialties provide care to patients.

At the micro level, the complexity and pace of change make care difficult. Healthcare is a rapidly developing field with over 700,000 publications produced annually and the pace in fact is increasing. (Smith et al. 2013). That there are already 14,400 codes in the World Health Organization's *International Statistical Classification of Diseases and Related Health Problems* (ICD) complicates the issue. Add to this the regulatory and administrative burden of primary care providers interacting with approximately 200 specialists in any given year and the complexity that care providers face on a daily basis becomes clear.

In short, the healthcare delivery system is itself a complex adaptive system and represents a Wicked_Problem_(glossary), whereby any changes to the system intended to solve an issue will likely create other issues.

Improving Ongoing Operations

As mentioned, above caregivers are faced with many challenges and the goal of in systems engineering in healthcare delivery is to lessen that burden in a systematic way without significant disruption of current operations. To do this successfully requires several factors:

- First, as stated above, systems engineers have to acknowledge that they are dealing with a complex adaptive system that includes many wicked problems. An analogy is that what systems engineers experience in healthcare is like rewiring a house with the power turned on because whatever changes are made are to an existing system that must operate while the changes are being made.
- Second, “the system” in place is difficult to define. The "healthcare system" is actually a combination of many open systems and interdependencies with the system of interest may be unknown.
- Third, patient safety is always a concern and any actions that could affect patient safety must be very carefully considered. Often, "optimizing" a system may introduce a potential risk to patient safety. These system aspects are always in tension.
- Fourth, there is a bias towards the current (known) system versus a change leading to an unknown system. Any change will create a certain amount of disruption to an operational system that may be currently operating at or beyond capacity.
- Fifth, healthcare delivery systems are combinations of patients, providers, process, and products and therefore uncertainty is a daily reality. This level of uncertainty may not be amenable to typical agile approaches of 4-6 week sprints nor traditional waterfall methods.
- Sixth, local factors could play a significant role; therefore no two sites may perform an operation in exactly the same way.
- Seventh, the entire industry acts as a complex adaptive system with multiple intelligent agents working sometimes in partnership and sometimes in conflict with the goals of the system or patient.

Because of these factors and others the tradition of healthcare systems engineering has been to use adaptable human-centered methods. (Checkland 1999)

History of Healthcare Improvement Research

There have been many attempts to understand and improve healthcare both in the public and private domains. Examples include the National Healthcare Service Change Model, the efforts of the Agency for Healthcare Research and Quality, and the Institute for Healthcare Improvement. Here we outline some representative efforts.

Healthcare improvement has been shaped in part by four seminal works by the Institute of Medicine (IOM). *To Err is Human* reported that up to 98,000 patients were killed by healthcare each year. (Kohn, Corrigan, & Donaldson 2000) This put an emphasis on safety as a key quality of care metric. The following year the Institute of Medicine (IOM) broadened the concept of quality beyond safety to include six measures of quality. They determined that healthcare should be safe, effective, patient centered, timely, efficient, and equitable. (Institute of Medicine 2001) This report called Crossing the Quality Chasm included an appendix that documented poor quality and the severity of the issues of under use, over use, and potential for harm in medicine. A search for the underlying reasons for poor quality led to three primary reasons for poor quality. The three reasons were the growing complexity of science and technology; the increase in chronic conditions; and the failure to exploit information technology.

To address these concerns the IOM partnered with the National Academy of Engineering (NAE) to see what could be done from a systems engineering perspective to address the real challenges facing the industry in Building a

Better Delivery System. (Compton et al. 2005). That was followed by the realization that standard systems engineering needed to be modified and healthcare was and would remain a human centered endeavor as stated in Best Care at Lower Cost (Smith et al. 2013)

Three Approaches

Although there are many accepted approaches to healthcare systems engineering and improvement, here we outline three that share common characteristics and are representative of most of the other methods.

The first approach is Lean Six Sigma which is a combination of two methods. Lean has its roots in the Toyota Production System (Ohno 1988) and the work of the International Motor Vehicle Program (Womack, Jones, & Roos 1990). Six-Sigma has its roots at Motorola and the work of Bill Smith. These two methods were combined by Michael L. George (see (George 2002) and (George 2003)). It includes techniques like value stream mapping, waste elimination, root cause analysis, and voice of customer. For additional information see Lean Engineering and Lean in Healthcare.

The second approach is based on industrial engineering, which has its roots in the work of Frederick Taylor and others. This approach includes tools such as discrete event simulation, ergonomics, production control, and operations research as shown in Figure 1. For additional information, see Systems Engineering and Industrial Engineering.

Insert Table ES-1 from Building a Better Delivery System here once we obtain the proper permissions.

The third approach is healthcare systems engineering. Traditional systems engineering uses a functional decomposition approach; see for example (Defense Systems Management College 2001). However, healthcare problems are often classified as wicked and complex and not amenable to traditional decomposition methods found in other areas of engineering. (Rouse & Serban 2014).

There are many tailored approaches to improving healthcare delivery, but almost all are based on one of these three approaches, or a combination of these.

Healthcare Systems Engineering

The basic systems engineering steps are similar to those for any industry specific applications, but the steps are tailored for healthcare. The traditional waterfall model of requirements, design, implementation, verification, and maintenance is interrupted in favor of almost continuous support. In many cases the closeout and transfer of the project to operational staff is more challenging in healthcare than in many industries.

Below is outlined a general methodology used by the US Department of Veteran's Affairs (VA) that may suit a wide variety of situations and programs, composed of 4 pillars: Define the Problem, Investigate Alternatives, Develop the Solution, and Launch and Assess the Solution. These 4 pillars are similar to classic mistake avoidance, development fundamentals, risk management, and schedule oriented approaches. (McConnell 1996); they are also similar to the Plan/Do/Check/Act methodology.

Define the Problem

As mentioned above, the patient is augmented by a care team consisting of family, friends, clinical staff, and many other support staff the patient will not directly encounter. This care team may not be familiar with the rigors of traditional engineering design. Because of this, a systems engineer may use a paired partnership model where engineers are embedded with clinical and administrative staff, family, and the patients themselves. In this concept, everyone is a designer and our goal is to provide them with the tools to contribute to the system design process. Even at this early stage, configuration management would be considered. Depending on the size of the rollout one alpha site and several beta sites may be used at any phase to avoid local optimal solutions that don’t work globally.

Investigate Alternatives

During the proof-of-concept phase, visualizing the result is important for the reasons mentioned above. Therefore, one or more initial prototypes may be developed with the alpha site. The goal is to get to a minimally viable product as soon as possible to demonstrate the viability of the product or methodology. After the initial conversations and meetings, participants have a need to have a common understanding of how the system will work. The systems engineer would embrace the concept of operations with rich pictures, model based systems engineering, story boards, customer journey maps and other tools so that we all have a common understanding of the proposed system.

Develop the Solution

Using what has been learned from the minimally viable product feedback and incorporate that into the future state optimization, one would continue developing the prototype at the initial paired partner alpha site and then the trusted beta demonstration sites. In our case, stakeholders are a part of the development team and not an ancillary function. For this reason, demonstration is considered a key element of the communication plan when developing the solution.

Launch the Solution and Access the Performance

During evaluation and deployment phase, a systems engineer would have considered the future state optimization with corresponding alpha and beta sites. Live implementation would then be used for further testing and evaluation. At any phase feedback is encouraged and reflected in the next iteration of the solution. As mentioned previously abandonment and closeout even during the live phase may not be practical and in fact could be disadvantageous because not all possible needed configurations or situations would have been encountered.

Example Systems Engineering Tools

Below is a list of systems engineering tools which could be used at each of the four steps.

1. Define the Problem
   1. Establish the scope and context of the problem (define boundary conditions)
   2. Stakeholder identification and management
   3. Lifecycle mapping
   4. Value Stream Process Mapping
   5. SWOT analysis (Operational Deficiencies and Technological Opportunities)
   6. Workflow/Usability/Use Case analysis
   7. Observation Research
   8. Root Cause Analysis (Fishbone diagrams, 5 whys, …)
2. Investigate Alternatives
   1. Requirements management
   2. SE Evaluation Methods (Decision Trees, Quality Function Deployment (QFD))
   3. Trade-off Analysis
   4. Model-Based Systems Engineering (MBSE)
   5. Technical Risk Management
3. Develop the Solution
   1. Concept Development
   2. Architecting the solution (functional analysis, subsystem decomposition, interface definition and control, modeling)
   3. Define the implementation
   4. Process Redesign Techniques (including Lean Six Sigma)
   5. Active Integration

   6. Agile / Lean Development Principles (iterative development)
4. Launch and Assess the Solution
   1. Managing Change in Organizations
   2. Stakeholder Management, Change Management Techniques
   3. Spiral, Agile, and Lean Startup Delivery Practices (Minimal Viable Product delivery)
   4. Business Risk Management
   5. Metrics and benchmarking

During all phases, elements of cognitive & organizational psychology, industrial engineering, usability engineering, systems engineering, and other facets may be critical to implement a solution. Humans are *the* major part of the system and even the system of systems approach in healthcare.

Conclusion

Systems Engineering for Healthcare delivery shares many aspects with traditional SE, but differs significantly since healthcare delivery is a service (not a product) and due to the domain specific challenges. In particular, problem definition is a particularly ‘wicked’ problem, and measuring successful outcomes in a clear and objective fashion is challenging.

References

Works Cited

Checkland, P. 1999. *Systems Thinking, Systems Practice*. Hoboken NJ: John Wiley.

Compton, W.D., G. Fanjiang, J.H. Grossman, & P.P. Reid. 2005. *Building a better delivery system: a new engineering/health care partnership*. Washington DC: National Academies Press.

Conover, C.J. 2012. *American health economy illustrated*. Washington DC: American Enterprise Institute.

Defense Systems Management College. 2001. *Systems engineering fundamentals: Supplementary text*. Fort Belvoir, VA: The Press.

George, M.L. 2002. *Lean Six Sigma*. New York, NY: McGraw-Hill.

George, M.L. 2003. *Lean Six Sigma for Service*. New York, NY: McGraw-Hill.

Griffith, J.R., & K.R. White. 2007. *The Well-Managed Healthcare Organization*. Chicago, IL: Health Administration Press.

Institute of Medicine. 2001. *Crossing the quality chasm: A new health system for the 21st century*. Washington DC: National Academy Press.

Kohn, L.T., J. Corrigan, & M.S. Donaldson. 2000. *To err is human: Building a safer health system*. Washington DC: National Academy Press.

McConnell, S. 1996. *Rapid Development*. Redmond, WA: Microsoft Press.

Ohno, T. 1988. *Toyota Production System*. Portland OR: Productivity Press.

PCAST. 2014. *Better Health Care and Lower Costs: Accelerating Improvement through Systems Engineering*. Washington DC.: President’s Council of Advisors on Science and Technology (PCAST).

Rouse, W.B., & N. Serban. 2014. *Understanding and managing the complexity of healthcare*. Cambridge, MA: The MIT Press.

Smith, M.D., R.S. Saunders, L. Stuckhardt, & J.M. McGinnis. 2013. *Best care at lower cost: The path to continuously learning health care in America*. Washington DC: National Academies Press.

Womack, J.P., D.T. Jones, and D. Roos.1990. *The machine that changed the world*. New York, NY: Harper Collins.

Primary References

PCAST. 2014. *Better Health Care and Lower Costs: Accelerating Improvement through Systems Engineering*. Washington DC.: President's Council of Advisors on Science and Technology (PCAST).

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Systems Biology

Systems biology is the computational and mathematical modelling of complex biological systems. Systems biology is a biology-based inter-disciplinary field of study that focuses on complex interactions within biological systems, using a holistic approach to biological research. From year 2000 onwards, the concept has been used in the biosciences in a variety of contexts. For example, the Human Genome Project is an example of applied systems thinking in biology which has led to new, collaborative ways of working on problems in the biological field of genetics. One of the outreaching aims of systems biology is to model and discover emergent properties of cells, tissues and organisms functioning as a system whose theoretical description is only possible using techniques which fall under the remit of systems biology. These typically involve metabolic networks or cell signalling networks. (Wikipedia Contributors 2016)

Systems Biology: A Vision for Engineering and Medicine

Organisms and Hosts Interact in Communities of Life

There is an increasing appreciation that microbes are an essential part of the ecologically-important traits of their host. Organisms do not live in isolation, but have evolved, and continue to evolve, in the context of complex communities and specific environmental conditions. Evolutionary biologists are increasingly able to integrate information across many organisms, from multiple levels of organization and about entire systems to gain a new integrated understanding that incorporates more and more of the complexity that characterizes interdependent species associations. Only when we begin to understand the molecular base for adaptation and interactions of communities of life, can we start to comprehend how ecosystems are functioning.

Addressing Different Levels of Organization of Organisms

Understanding the function of complex biological systems is one of the greatest challenges facing science. The rewards of success will range from better medicines to new engineering materials. The sequencing of the human genome, although of fundamental importance, does not even provide a complete parts list of the protein molecules that exist in a biological organism because of complexities of downstream processing and complex folding required to make a functioning receptor or enzyme from a long chain of amino acids. Furthermore, protein molecules do not function alone but exist in complex assemblies and pathways that form the building blocks of organelles, cells, tissues, organs, organ systems and organisms, including man. The functioning of brain or muscle, liver or kidney, let alone a whole person, is much greater than the sum of its parts.

Figure 1 - Levels of Structural Organization of the Human Body (source - https:/ / cnx. org/ contents/ Xh_25wmA@7/Structural-Organization-of-the#fig-ch01_02_01)'"

Internalizing the Complexity – Pushes the Boundary of Systems Thinking Capability

To tackle this problem (understanding biological systems) requires an iterative application of biomedical knowledge and experiment with mathematical, computational and engineering techniques to build and test complex mathematical models. Systems and control engineering concepts, a modular approach and vastly increased computing capacity are of critical importance. The models, once developed and validated, can be used to study a vastly greater range of situations and interventions than would be possible by applying classical reductionist experimental methods that usually involve changes in a small number of variables. This new approach is now termed "Systems Biology". It allows insight into the large amount of data from molecular biology and genomic research, integrated with an understanding of physiology, to model the complex function of cells, organs and whole organisms, bringing with it the potential to improve our knowledge of health and disease. Systems Biology will inevitably become an approach that pervades scientific research, in much the same way that molecular biology has come to underpin the biological sciences. It will transform the vast quantities of biological information currently available into applications for engineering and medicine.

Natural Patterns and Engineered Patterns Can Be a Source of Inspiration - in Both Directions

Biological organisms are much more complicated than any machine designed by man. However, there are similarities between the way in which organs and whole organisms are assembled from molecules and cells and the design methods used by engineers in the construction of complex systems. The application of such methods to biology will, however, require novel engineering tools to be developed since biological systems possess key features that artificial ones do not. Specifically, biological systems have an exceptional capacity for self-organization and assembly, using rules and mechanisms that have been shaped by natural selection. Biological systems also have significant capacity for continuing self-maintenance through turnover and renewal of component parts. Perhaps the property that distinguishes biological systems most is their ability to auto-adapt their organization to changing circumstances through altered gene expression, or more directly, through signal transduction and modification of proteins. This adaptation culminates at higher levels of organization as evidenced by phenomena such as the development of resistance to antibiotic therapy or tolerance to recreational drugs. The mechanisms by which component parts interact are often highly stochastic in nature; that is, susceptible to the play of chance, which becomes particularly important when only a few components are being considered. Nevertheless, biological systems are robust.

Advancements in Methods for Predicting “What If” in the Behavior of Complex Adaptive Systems

Advances in engineering design and techniques carry a significant potential in driving the progress of Systems Biology. Interventions to biological systems intended to improve health, whether environmental, pharmacological or clinical, need to be carefully thought through and carried out to maximize benefit and reduce harm. The refinement of techniques and tools enables devices and systems to achieve a defined performance within precise tolerance limits, potentially allowing better interventions to complex biological systems. They will be increasingly necessary to permit more reliable system-wide predictions of the effects of biomedical advances and to achieve desired clinical results to a predefined tolerance, or at least to have a quantitative bound on the biological uncertainty.

Transdisciplinary Approaches are Needed to Address the Complex Bio-system Problems

Research in the field of Systems Biology requires close interactions and collaborations between many disciplines that have traditionally operated separately such as medicine, biology, engineering, computer science, chemistry, physics and mathematics. Systems Biology demands a focus on the problem as a whole and therefore a combination of skills, knowledge and expertise that embraces multiple disciplines. The success of leaders in the field of Systems Biology will depend strongly on the extent to which they accomplish the creation of the environment that researchers need to develop an understanding of different working cultures, and manage also to implement strategies that integrate these cultures into shared working practices.

Systems Biology: Relevance to Healthcare

Complex Diseases Demand Systemic Approaches

Over the past few decades, pharmaceutical R&D has focused on creating potent drugs directed at single targets. This approach was very successful in the past when biomedical knowledge as well as cures and treatments could focus on relatively simple causality. Nowadays, the medical conditions that affect a significant proportion of the population in industrialized countries are more complex, not least because of their multifactorial nature. The sequencing of the human genome has led to a considerable increase in the number of potential targets that can be considered in drug discovery and promises to shed light on the etiology of such conditions. Yet, the knowledge of the physiological properties and the role that these targets play in disease development is still limited.

Diminishing Returns in the Single Target Approach to Disease

In terms of drug targets, there is a case that much of 'the low hanging fruit' was picked in the period between the late 1940s and the mid-1980s. The decline in output of new molecular entities and medicines recorded over the last 20 years, despite the steadily growing R&D expenditure and significant increase in sales, bears testimony to the fact that advances with new targets are more difficult and that R&D projects have become much more prone to failure. A basic problem is that the many factors that predispose to, and cause, complex diseases are poorly understood let alone the way in which they interact. The very fact that there are multiple drivers for these conditions suggests that a reductionist approach focusing on individual entities in isolation is no longer appropriate and may even be misleading. It is therefore necessary to consider 'novel' drugs designed to act upon multiple targets in the context of the functional networks that underlie the development of complex diseases. Many of the new developments are likely to turn into effective medicines when combinations of drugs are used to exert a moderate effect at each of several points in a biological control system. Indeed, many common diseases such as hypertension and diabetes are already treated with a combination of two or three medicines hitting different targets in the control network that underlies the condition. Investigating the possible combinations by trial and error in man is onerous but feasible with two components. However, it quickly becomes extremely complicated with three components and well-nigh impossible with four or more. Systems Biology, promises to assist in the development of more specific compounds and in the identification of optimal drug targets on the basis of their importance as key 'nodes' within an overall network, rather than on the basis of their properties as isolated components.

Individualized Medicine, Tuned to the Individual and Their Circumstances

Increasingly powerful drugs will be aimed at a decreasing percentage of people and eventually at single individuals. Modelling can be used to integrate in vivo information across species. Coupled with in vitro and in silico data, it can predict pharmacokinetic and pharmacodynamics behavior in humans and potentially link chemical structure and physicochemical properties of the compound with drug behavior in vivo. Large-scale integrated models of disease, such as diabetes and obesity, are being developed for the simulation of the clinical effects resulting from manipulations of one or more drug targets. These models will facilitate the selection of the most appropriate targets

and help in planning clinical trials. Coupling this approach with pertinent genomic information holds the promise of identifying patients likely to benefit most from or to be harmed by, a particular therapy as well as helping in the stratification of patients in clinical trials. Symptoms that diagnose a disease do not necessarily equate to a common cause.

Systems Biology is arguably the only research approach that has the potential to disentangle the multiple factors that contribute to the pathogenesis of many common diseases. For example, hypertension, diabetes, obesity and rheumatoid arthritis are known to be polygenetic in origin although individual genes may not have been identified. Ultimately, the prevention of these conditions rests upon a comprehensive approach that engages with each of the more important predisposing factors, genetic and environmental, that operate upon individuals. A systems approach is already proving valuable in the study of complex scientific subjects and the research aimed at the prevention and management of medical conditions. Illustrative examples are neuroscience, cancer, ageing and infectious diseases.

A Healthcare Paradigm Reinforcing the Causes of Health and Not Just the Treatment of Disease

Notwithstanding the hugely important role that Systems Biology plays in understanding disease and designing drugs that treat them, the greatest opportunities may lie in health maintenance and disease prevention. Even modest measures that could retard the effect of ageing on brain, heart, bones, joints and skin would have a large impact on the quality of life and future healthcare demands of older people and consequently on the provision of health services. Young people are vulnerable too. Multifactorial diseases such as diabetes, obesity, allergies and autoimmune conditions are becoming prevalent in younger people and unless effective measures are taken to prevent an early and significant decline in their health, healthcare demand will increase exponentially. It is apparent that multiple and diverse factors interact in determining health, quality of life and ageing. These include genetic makeup, microbiota, diet, physical activity, stress, smoke and alcohol, therapeutic and social drugs, housing, pollution, education, and only a systems approach will permit the understanding of how best to prevent and delay health decline.

References

Works Cited

Wikipedia contributors. "Systems biology." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 20 Aug. 2016. Web. 12 Sep. 2016.

Primary References

None.

Additional References

Bosch, T.C.G. and M.J. Mc-Fall-Ngai. 2011. "Metaorganizsms as the new frontier." *Zoology.* 114(4): 185-190. September 2011. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3992624/

Dollery, C. and R. Kitney. 2007. *Systems Biology: a vision for engineering and medicine.* London: The Academy of Medical Sciences and The Royal Academy of Engineering. Available at https://www.acmedsci.ac.uk/viewFile/publicationDownloads/1176712812.pdf

Endy, D. 2005. "Foundations for engineering biology." *Nature*. 438(7067): 449-453. 24 November 2005. Available at: http://www.nature.com/nature/journal/v438/n7067/abs/nature04342.html

Harvard Medical School. 2010. "Department of Systems Biology." Cambridge, MA: Harvard Medical School, Harvard University. Available at: https://sysbio.med.harvard.edu/

Kitano, H. 2002. "Systems Biology: A Brief Overview." *Science.* 295(5560): 1662-1664. 01 March 2002. Available at: http://science.sciencemag.org/content/295/5560/1662.

Sauser, B., J. Boardman, and D. Verma. 2010. "Toward a Biology of Systems of Systems." *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans.* 40(4): 803 - 814. Available at: http://ieeexplore.ieee.org/document/5467221/

Wikipedia contributors. "Systems biology." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 20 Aug. 2016. Web. 12 Sep. 2016.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Lean in Healthcare

Lean Thinking, or Lean for short, originated in Toyota factories in the 1960s, was "transplanted" to the U.S. in 1992 with the publication of Womack and Jones' *Lean Thinking: Banish Waste and Create Wealth in Your Corporation* (2003), and evolved globally to practically all work domains: healthcare, engineering and systems engineering, science, administration, supply chain, government, banking, aviation, and many others (Oppenheim 2011). Lean has proven itself as the most effective methodology for improving operations identifying and eliminating waste from work processes. (E.g. Womack and Jones 2003; Oppenheim 2011; Graban 2012; Toussaint and Gerard 2010; and Oehmen 2012) Since 2003, Lean has established itself in healthcare operations.

Overview of Lean in Healthcare

Entire medical organizations (e.g., Theda Care, WI; Jefferson Healthcare, WA; Virginia Mason, WA; Geisinger Health (now called ProvenCare), PA; St. Elizabeth, Tilburg, The Netherlands, and numerous others (e.g. Graban 2012; Toussaint and Gerard 2010)) have been transformed with Lean. These sources contain rich data on specific improvements. Most leading healthcare institutions now have Lean centers of excellence or use Lean consultants, including Kaiser Permanente, Mayo Clinic, UCLA, Veterans Administration, and others. Lean has proven itself in reducing turnaround time of clinical tests, the time spent by patients in emergency departments, operating suites, pharmacies and clinics. Lean improvements in healthcare on the order of 30-50% are routine because traditional healthcare operations are burdened with this much waste, which remains "unseen" by the employees unless they are trained in Lean. Lean is now an established paradigm for improving healthcare delivery operations: increasing quality of healthcare, delivering care faster, shortening patient time in the system, increasing the time of medical professionals with the patient, reducing bureaucracy, increasing capacity of operations, and reducing healthcare costs and frustrations. (Graban 2012; Toussaint and Gerard 2010)

Lean does not mean that people have to work faster or "attach roller blades to move around faster". In Lean, systems employees work at their regular ergonomic and intellectual speeds. The time savings come from finding and eliminating idle states (e.g., waiting in numerous queues in the emergency departments), reduction of mistakes and rework, elimination of non-value adding tasks, and more streamlined movements of patients, staff, equipment, and supplies. And, most emphatically, Lean does not mean "mean layoffs". Quite the opposite is true: Lean improves human relations at work and changes the culture from the traditional "blaming and shaming" to teamwork and cooperation focused on the good of the patient. (Graban 2012 (in particular see the endorsements from eight medical professionals on pages ii and iii) and (Toussaint and Gerard 2010)

With the endorsement of *Lean for Systems Engineering with Lean Enablers for Systems Engineering in the Wiley Series*, (Oppenheim 2011) the International Council on Systems Engineering (INCOSE) has effectively adopted Lean as one of its essential competencies. This book was followed with a major joint Project Management Institute

(PMI)-INCOSE-MIT publication of (Oehmen 2012) integrating Lean with Systems Engineering and Program Management. Indeed, when applied with Systems Engineering and Systems Thinking, Lean becomes a powerful weapon in bending the healthcare cost curve and improving the quality of care.

Three concepts are critical to the understanding of Lean: value, waste, and the process of creating value without waste, which has been captured into the so-called Six Lean Principles, as follows.

- **Value**: M. Porter (2010) suggested that patients value three levels of care: (1) survival and the degree of recovery; (2) the time required to get back to normal activities, and (3) the sustainability (individual and social cost) of treatments.
- **Waste**: Table 1 lists the eight categories of waste used in healthcare. (Graban 2012; Toussaint 2010)

Table 1. Eight Waste Categories Used in Lean Helathcare (SEBoK Original).

Waste Type	Healthcare Examples
1. Waiting	Patients wait in numerous queues in clinics, test facilities, ERs, pharmacies, and for insurance approvals; MDs wait for next activity to occur (e.g. test results, information, approvals.)
2. Over-processing	Performing work that is not valued or needed, e.g. MDs and RNs spending time on computer filling out bureaucratic forms that nobody will review.
3. Over-production	Performing more work than needed for value. Transport of a patient in a wheelchair performed by expensive medical professionals because of the lack of transporters.
4. Inventory	Excess inventory costs. Expired supplies that must be thrown away.
5. Transportation of Patients	Transportation of patients over long distances to test offices in hospitals. Poor layout of hospitals, EDs, or test facilities.
6. Motion of Staff	Staff walking over long distances to fetch supplies, and between patients and central hospital stations.
7. Defects	Treatment of hospital infections. Failed and repeated tests, repeated paperwork. Surgical cart missing an item. Wrong medicine.
8. Waste of Human Potential	Burnout of medical staff. Frustrated employees quit making suggestions for improvements.

Table 2 lists the six Lean Principles (Graban 2012) and provides healthcare examples.

Table 2. Six Lean Principles (SEBoK Original)

Principle Name	Explanation
1. Value	Specify value from the perspective of the customer: the patient.
2. Value Stream	Identify all the value-added steps across the entire process, crossing all departmental boundaries, linking the steps into a seamless process, and eliminating all steps that do not create value.
3. Flow	Keep the processes flowing smoothly through all the steps, eliminating all causes of delay, such as batches of patients or items, and quality problems.
4. Pull	Avoid pushing work onto the next step or department; let work and supplied be pulled, as needed, when needed.
5. Perfection	Pursue perfection through continuous improvement, Kaizen events, implement best work standards, checklists, training, and promote improvement teams and employee suggestions.
6. Respect People	Create work environment based on synergy of cooperation, teamwork, great communication and coordination. Institute leadership. Abandon the culture of blaming and shaming.

Lean Practices

Lean healthcare strongly promotes engaging and leading employees. Lean places a big value on continuous education and training of employees at all levels. Lean management promotes standardization of best practices ("the best known way of doing it", but not necessarily "identical"), checklists, redundancies, patient safety and privacy rules, and patient data security and cybersecurity. Lean advocates visual management, with electronic or "black" boards updated in real time and displaying all information important for the local employees to manage their operation efficiently. Patient safety is still a significant problem in the U.S., in 1999 causing almost 250,000 deaths (Institute of Medicine, 1999) and medical errors occur in one of three admissions. Instead of "blaming and shaming" Lean promotes error and harm prevention and deep root-cause analysis, implementing processes and tools that make it impossible to create an error.

Systems Thinking and Lean

Healthcare is the most complex socio-technological system in our society, consuming nearly 20% of the U.S. GDP. Healthcare should be safe, effective and evidence based (Berwick 2011), as well as affordable and accessible, efficient, patient centered, timely, well integrated, and inclusive of latest science. (Oppenheim 2015) Healthcare has many stakeholders: the patients, medical professionals, medical facilities, hospitals, clinics, labs, medical equipment makers and users, pharmaceuticals, healthcare researchers and academia, insurances, employers, federal & state governments and international disease prevention centers, military and veteran's administration, fire departments and ambulances and others. The number of potential interactions (interfaces) in this hyper-system is extensive, and many interfaces are nonlinear, "wicked" (interacting with unpredictable humans), often creating unintended consequences and emergent behaviors. Because of these vast complexities, healthcare leaders (e.g. Kanter 2015) point out the need for intensive application of systems thinking and Lean when addressing these challenges. Attempting to solve the complex healthcare problems without systems thinking risks myopic and unsafe attempts which create more problems than they solve. Attempting to solve the challenges without Lean inevitably promotes excessive wastes, costs, and inefficiencies. Good healthcare needs both, Systems Thinking and Lean, to be applied simultaneously.

Lean and Agile in Six Healthcare Value Streams

The Healthcare Working Group of INCOSE identified six following value streams for HSE: A. Systems Engineering for medical devices B. Systems Engineering for healthcare informatics and medical records C. Healthcare delivery (operations) D. Biomedicine and big data analytics E. Pharmaceutical value streams F. Healthcare public policy

As described above, Lean is extraordinarily effective and well established in improving healthcare delivery operations (C). Agile is highly effective in (B) because this value stream works with software, the domain from which Agile originated. Since the stream (A) is the most similar to traditional systems engineering, Agile is expected to be effective therein, although Agile is not yet highly popular in healthcare outside of the software domain. Elements of Lean improvements which are localized and weakly convoluted (e.g., Kaizen events) have strong overlap with Agile/Scrum methodology. (Medinilla 2014)

MBSE and Lean

A highly powerful Model Based Systems Engineering (MBSE) is clearly the tool of choice for those applications where the benefit from multiple use of a standardized (reference) architecture and standard model compensates for the significant effort of creating such a model or architecture. (OMG 2016) In healthcare the value streams (A), (B), (E) and potentially F are the most conducive to the application of MBSE. Lean thinking is applicable to any healthcare operation without limitation. The Lean improvements always begin with the so-called Gemba waste walks, during which experts together with local process stakeholders walk along all the process steps, interviewing stakeholders and identifying and measuring the wastes wherever they occur. The rich menu of Lean thinking

processes and tools is then applied to eliminate the wastes. Training and active participation of local stakeholders is always required.

Examples of Lean Improvements

1. In Jefferson Healthcare, WA: (Murman 2010)
    - In Acute Myocardial Infarction (a severe heart attack) time is critical as the greatest loss of heart muscle is in the first two hours. Recommended treatment is catheter insertion of balloon within 90 min of the contact with the patient (wherever the patient happens to be located). The Lean approach has reduced the treatment time from 165 min to 20-60 min at the patient site, vastly increasing patient survival rate.
    - The five Jefferson Healthcare clinics increased the cumulative available clinic hours from 1400 to 5600 in two years of Lean improvements which were focused on reorganizing medical staff schedules and eliminating wasted times, with no staff additions. The available clinic hours directly translate into billable visits: 1175 additional patients have been seen in 2009 compared to 2008 across the five clinics.
    - The Operating Room daily "on time start" of actual operations went from 14% to 96% using Lean tools for process planning and workplace organization.
    - Harder to measure is the culture change, although the staff participation at Lean improvement events was at 50%.
2. In Kaiser Permanente Southern California: (Oppenheim 2015)
    - In nine regional clinical laboratories Lean improvements cut the turnaround time for laboratory results by between 30 and 70%, with significant corresponding reductions of cost, rework, errors and work morale, and without hiring new staff or adding equipment.
    - In two Emergency Departments (ED) the average patient length of stay was reduced by 40% by the elimination of various idle states. The ED capacity increased accordingly.
    - The amount and cost of inventory of supplies on hand was reduced by nearly 30% by introducing the Just-in-Time tools of Lean.
3. In Alegent Health, NE (Graban 2012) the turnaround time for clinical laboratory results was reduced by 60% in 2004 without adding new staff or equipment; and by another 33% from 2008 to 2010.
4. In Kingston General Hospital, Ontario (Oehmen) the instrument decontamination and sterilization cycle time was reduced by 54% while improving productivity by 16%.
5. In Allegheny Hospital, PA the central-line associated bloodstream infections were reduced by 76%, reducing patient death from such infections by 95% and saving $1 million.
6. In UPMC St. Margaret Hospital, PA (Graban 2012) the readmission rates for chronic obstructive pulmonary disease (COPD) patients were reduced by 48%.
7. In ThedaCare, WI [3] the waiting time for orthopedic surgery was reduced from 14 weeks to 31 hours (from first call to surgery); improved inpatient satisfaction scores of "very satisfied" rose from 68% to 90%.
8. In Avera McKennan, SD [3] the patient length of stay was reduced by 29%, and $1.25 million in new ED construction was avoided.
9. In Denver Health, CO [3] the bottom-line Lean benefit was increased by $54 million through cost reduction and revenue growth, and layoffs were avoided.
10. In Seattle Children's Hospital, WA $180 million in capital spending was avoided through Lean improvements.

These examples demonstrate that Lean is successful in cost and throughput time reductions, and improvements in quality and patient and staff satisfaction. The improvements of this level are possible, even routine – because the amount of initially-invisible waste in traditional healthcare organizations is so high. The broad range of operations described in the examples manifest that Lean is applicable across the board to healthcare operations, without limitations.

Education in Lean Healthcare

Increasingly, Lean Healthcare becomes an inherent part of Healthcare Systems Engineering (HSE) Master's Programs, e.g. (Loyola Marymount University 2016) which has been developed in collaboration with Kaiser Permanente. The program includes two courses in Lean, basic and advanced, focused on improving operations in clinics, hospitals, emergency departments, clinical laboratories, radiology testing, operating rooms, pharmacies, supply chain, and healthcare administration. After the basic courses in systems engineering, project management, and systems thinking, the students also take courses on healthcare system architecting, modeling and simulations; medical data mining and analytics; systems engineering for medical devices, healthcare enterprise informatics; and healthcare delivery systems. All these advanced courses contain elements of Lean thinking because all these subdomains risk being burdened with waste and poor quality if Lean is ignored. Simply put, Lean is not really an optional extra if you want to achieve efficiency and effectiveness.

References

Works Cited

B.W. Oppenheim, B.W. 2015. "Lean Healthcare," INCOSE Healthcare Working Group webinar. San Diego, CA: International Council on Systems Engineering. April 30, 2015. Available at: https:/ / onedrive. live. com/ redir?resid=147E5C4249DA0EFB%21142

Berwick, D. 2009, "National Forum Keynote, Institute for Healthcare Improvement." Cambridge, MA: Institute for Healthcare Improvement. Available at: http:/ / www. ihi. org/ IHI/ Programs/ AudioAndWebPrograms/ BerwickForumKeynote2009.htm (accessed July 4, 2011)

Graban, M. 2012. Lean Hospitals; Improving Quality, Patient Safety, and Employee Engagement. Boca Raton, FL: CRC Press.

Kanter, M.K. 2015. "Strategic Partnership of Healthcare and Systems Engineering." San Diego: INCOSE Healthcare Working Group presentation, 2015

Loyola Marymount University. 2016. "MS Degree Program in Healthcare Systems Engineering." Available at: CSE.lmu.edu/graduateprograms/systemsengineering/healthcaresystemsengineeringms/

Medinilla, Á. 2014. ''Agile Kaizen: Managing Continuous Improvement Far Beyond Retrospectives.'' New York, NY: Springer, 2014

Murman, E. 2010 "The Lean Aerospace Initiative." Boston MA: Lean Advancement Initiative (LAI) Annual Conference.

Oehmen, J. 2012. The Guide to Lean Enablers for Managing Engineering Programs. PMI-INCOSE-LAI MIT. May 2012.

OMG. 2016. "MBSE Wiki." Available at: http://www.omgwiki.org/MBSE/doku.php (last accessed March 29, 2016)

Oppenheim, B.W. 2011. Lean for Systems Engineering with Lean Enablers for Systems Engineering. Hoboken, NJ: Wiley Series in Systems Engineering and Management.

Porter, M. 2010. "What is Value in Healthcare?" New England Journal of Medicine. 363: 2488-2481. 08 December 2010.

Toussaint, J. and R. Gerard. 2010. On the Mend: Revolutionizing Healthcare to Save Lives and Transform the Industry. Cambridge, MA: Lean Enterprise Institute. 06 June 2010.

Womack, J.P. and D. T. Jones. 2003. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Washington, DC: Free Press.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article (Part 5) >

SEBoK v. 1.9.1, released 16 October 2018

Part 5: Enabling Systems Engineering

Enabling Systems Engineering

Part 5 of the Guide to the SE Body of Knowledge (SEBoK) is a guide to knowledge about how an enterprise prepares and positions itself to effectively perform the systems engineering (SE) activities described elsewhere in the SEBoK.

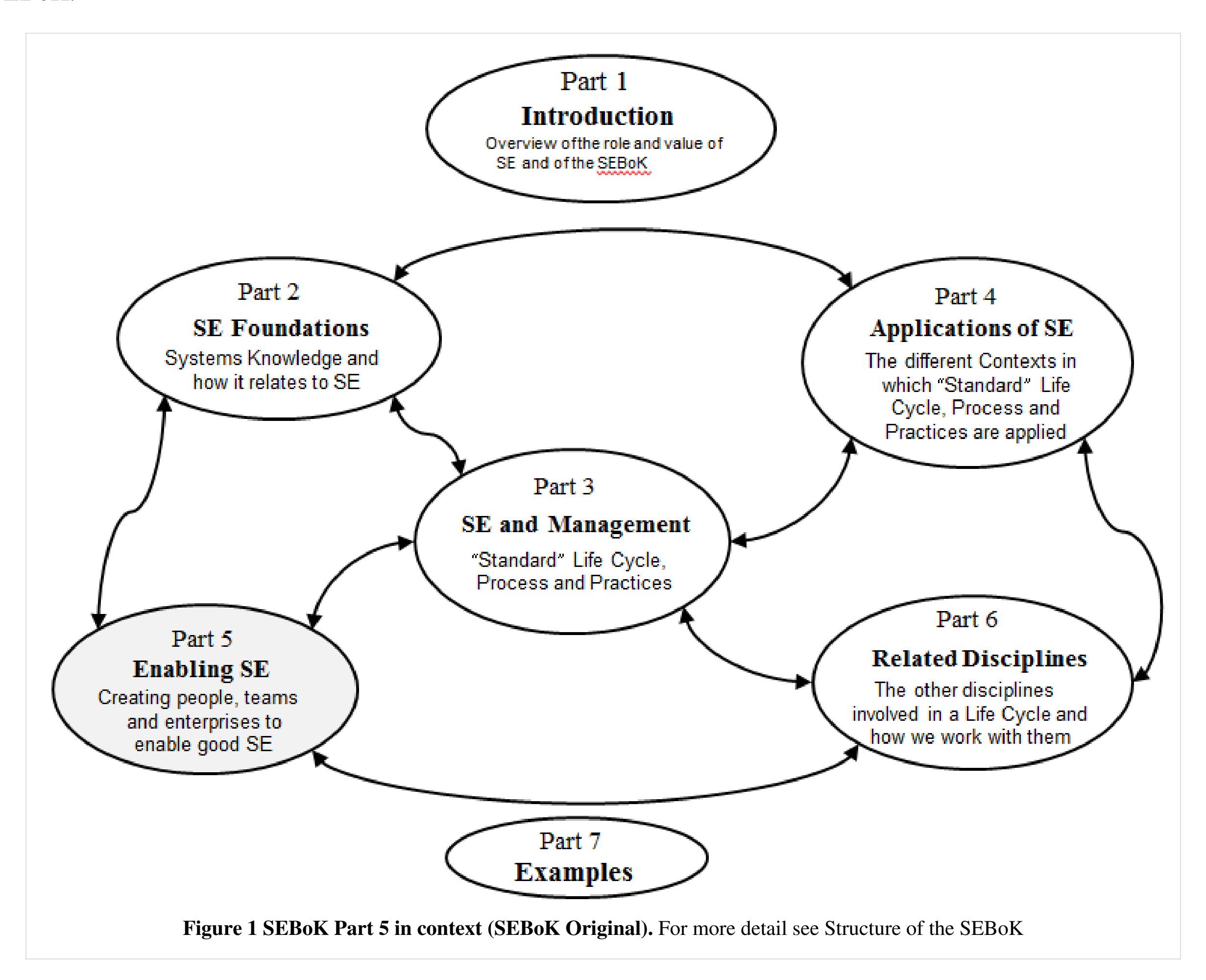


Figure 1 SEBoK Part 5 in context (SEBoK Original). For more detail see Structure of the SEBoK

SE activities—how to develop requirements, select an appropriate life cycle model, and architect a system of systems, and so on—are covered elsewhere, especially in Part 3, Systems Engineering and Management. An organization that desires to do these things effectively must work through questions like whether to allow a project manager to select the systems engineers he or she employs, and, if so, what competencies the project manager might seek in those systems engineers. These are the kinds of questions that Part 5 explores.

The discussion defines three levels of organization: enterprise or business, team, and individual. To adapt an example to a more complex organizational structure, simply decompose enterprises into sub-enterprises and teams into sub-teams, as needed. For more about the different types of enterprises, see Types of Systems in Part 2.

Knowledge Areas in Part 5

Each part of the SEBoK is composed of knowledge areas (KA). Each KA groups topics around a theme related to the overall subject of the part.

The KAs in Part 5 explore how to enable an organization to perform SE:

- Enabling Businesses and Enterprises
- Enabling Teams
- Enabling Individuals

Common Practices

There are as many different ways to enable SE performance as there are organizations, and every organization's approach is detailed and unique. Nevertheless, common practices, methods, and considerations do exist. Part 5 uses them as a framework to structure the relevant knowledge.

SE activities that support business needs and deliver value are enabled by many factors, including

- Culture (see Culture),
- SE competencies (see Determining Needed Systems Engineering Capabilities in Businesses and Enterprises) and how the organization grows and deploys its workforce to acquire them, and
- SE tooling and infrastructure (see Systems Engineering and Management in Part 3).

Enterprises and Businesses

The fact that Part 5 uses two terms, “Enterprise” and “Business,” to name a single level of organization, indicates that the two are closely related. In many contexts it is not necessary to make any distinction between them: an enterprise may be a traditional business, and a business can be seen as a special type of enterprise. For the sake of brevity, the term “business” is used to mean “business or enterprise” throughout most of Part 5.

Traditional businesses usually have a legal structure and a relatively centralized control structure. Such a business may be a corporation, or a unit of a company or government agency, that creates a product line or offers services.

On the other hand, an enterprise can be structured in a way that excludes description as a business. This happens when the enterprise crosses traditional business boundaries, lacks a centralized legal authority, and has relatively loose governance. One example is the healthcare system in the US which encompasses hospitals, insurance companies, medical equipment manufacturers, pharmaceutical companies, and government regulators. Another is the set of companies that form the supply chain for a manufacturer, such as the thousands of companies whose parts and services Apple uses to create, distribute, and support the iPhone.

Significant actions that enable SE are often conducted by traditional businesses rather than by less tightly-structured enterprises. Even so, organizational context affects how the business approaches SE and therefore how it enables SE performance. A business that sells to the general commercial marketplace typically has far fewer constraints on its SE practices than one which performs contract work for a government agency. A business that creates systems with very demanding characteristics, such as aircraft, typically has a much more rigorous and planned approach to SE than one which creates less demanding systems, such as an app for a smartphone.

Traditional businesses are intended to be permanent, and typically offer a portfolio (glossary) of products and services, introduce new ones, retire old ones, and otherwise seek to grow the value of the business. Sometimes a single product or service has such value and longevity that it spawns a business or enterprise just for its creation, maintenance, and support. The Eurofighter Typhoon aircraft, for example, was developed by a consortium of three corporations that formed a holding company specifically to provide support and upgrade services throughout the in-service life of the aircraft.

For more on the distinction between businesses and enterprises and the value of systems engineering of enterprises to them, see Enterprise Systems Engineering in Part 4. Systems of Systems (SoS), also in Part 4, contrasts the tighter control over SE that is usual for businesses with the looser control that is usual for enterprises lacking a traditional business structure. Groupings of Systems in Part 2, discusses the Directed SoS, to which the traditional business may be equivalent.

Teams

Teams operate within the context of the businesses in which they reside. This context determines how the team is enabled to perform SE.

For example, a business may grant a team wide autonomy on key technical decisions, which are made either by systems engineers on the team or in consultation with team systems engineers. On the other hand, the same business could instead create a generic set of SE processes that all teams are to tailor and use, constraining the team to adhere to established business policies, practices, and culture. The business could even require that the team gain approval for its tailored SE process from a higher level technical authority.

Teams are usually formed for a limited duration to accomplish a specific purpose, such as creating a new system or upgrading an existing service or product. Once the purpose has been fulfilled, the team responsible for that effort is usually disbanded and the individuals associated with the effort are assigned to new tasks. Exceptions do happen, however. For example, a team of systems engineers tasked with assisting troubled programs throughout a corporation could persist indefinitely.

References

Works Cited

None.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Enabling Businesses and Enterprises

Enabling Businesses and Enterprises

Part 5 on Enabling Systems Engineering explores how systems engineering (SE) is enabled at three levels of an organization: the business or enterprise (hereafter usually just called "business" --- See Enabling Systems Engineering for more on this), the team, and individuals.

The **Enabling Businesses and Enterprises** Knowledge Area describes the knowledge needed to enable SE at the top level of the organization. Part 3, Systems Engineering and Management, describes how to perform SE once it has been enabled using the techniques described in Part 5. Moreover, a business is itself a system and can benefit from being viewed that way. (See Enterprise Systems Engineering in Part 4.)

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Systems Engineering Organizational Strategy
- Determining Needed Systems Engineering Capabilities in Businesses and Enterprises
- Organizing Business and Enterprises to Perform Systems Engineering
- Assessing Systems Engineering Performance of Business and Enterprises
- Developing Systems Engineering Capabilities within Businesses and Enterprises
- Culture

Relationship Among Topics

- Systems Engineering Organizational Strategy describes how SE delivers value to the business, who makes decisions about SE in the business, how are those decisions made, how resources are allocated, and how the soundness and performance of those decisions are monitored.
- Determining Needed Systems Engineering Capabilities in Businesses and Enterprises describes how a business decides what specific SE capabilities are needed; e.g., a business that creates cutting edge products would likely require very strong architecting capabilities, including modeling tools. A business that has a global development team would likely need a very robust collaboration toolset.
- Organizing Business and Enterprises to Perform Systems Engineering describes various organizational models; e.g., which SE functions should be centralized, which should be distributed, how much SE every engineer should know.
- Assessing Systems Engineering Performance of Business and Enterprises describes how a business understands how well it is doing with respect to the SE actually being performed using the techniques described in Systems Engineering and Management.
- Developing Systems Engineering Capabilities within Businesses and Enterprises describes how SE talent that delivers the desired SE capabilities is grown and acquired
- Finally, Culture describes how the culture of a business affects SE; e.g., a risk-averse business will likely use plan-driven SE processes; an entrepreneurial fast-pace business will likely use agile SE processes (See Life Cycle Models).

To some extent, these topics have the character of a "plan-do-check-act" cycle, where the "do" part of the cycle is performing SE using the techniques described in Part 3, Systems Engineering and Management(Deming Part 3). For example, if assessing the business' SE performance shows shortfalls, then additional SE capabilities may need to be developed, the organization may need to be adjusted, processes may need to be improved, etc., all working within the existing cultural norms. If those norms prevent the business from successfully performing SE, then transformational efforts to change the culture may be needed as well.

References

Works Cited

Deming, W.E. 1994. *The New Economics*. Cambridge, MA, USA: Massachusetts Institute of Technology, Centre for Advanced Educational Services.

Primary References

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*, 3rd ed. Hoboken, NJ, USA: John Wiley and Sons.

Elliott, C. et al. 2007. *Creating Systems That Work – Principles of Engineering Systems for The 21st Century.* London, UK: Royal Academy of Engineering. Accessed September 2, 2011. Available at http://www.raeng.org.uk/education/vps/pdf/RAE_Systems_Report.pdf.

Hofstede, G. 1984. *Culture's Consequences: International Differences in Work-Related Values*. London, UK: Sage.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College, UK.

Morgan, J. and J. Liker. 2006. *The Toyota Product Development System: Integrating People, Process and Technology*. New York, NY, USA: Productivity Press.

Rouse, W. 2006. *Enterprise Transformation: Understanding and Enabling Fundamental Change.* Hoboken, NJ, USA: John Wiley and Sons.

Senge, P. M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization,* 2nd ed. New York, NY, USA: Currency Doubleday.

Shenhar, A.J. and D. Dvir. 2007. *Reinventing Project Management: The Diamond Approach to Successful Growth and Innovation*. Boston, MA, USA: Harvard Business School Publishing.

Additional References

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Systems Engineering Organizational Strategy

Virtually every significant business (glossary) or enterprise (glossary) that creates products (glossary) or services (glossary) benefits from performing a wide variety of systems engineering (SE) activities to increase the value (glossary) that those products and services deliver to its owners, customers, employees, regulators, and other stakeholders. (See Stakeholder Needs and Requirements.)

A business is a specific type of enterprise, usually a legal entity with a management structure that allows for relatively tight control of its components...including how it enables SE. The term business is often used in this article in lieu of enterprise because specific actions to enable SE are typically done by businesses. This is discussed further in the parent article Enabling Systems Engineering. The strategy for organizing to conduct SE activities is important to their effectiveness. For example, every enterprise has a purpose, context, and scope determined by some of its stakeholders and modified over time to increase the value the enterprise offers to them.

Some enterprises are for-profit businesses. Others are not-for-profit businesses that work for the public good. Still others are non-traditional businesses, but more loosely structured entities without legal structure, such as a national healthcare system. Some enterprises are located at a single site, while some others are far-flung global "empires". Some work in highly regulated industries such as medical equipment, while others work with little government oversight and can follow a much wider range of business practices. All of these variations shape the strategy for performing SE.

Primary Considerations

SE organizational strategy is driven by the goals of the business and the resources and constraints available to achieve those goals. SE strategy in particular is influenced by several considerations:

- The purpose of the business
- The value the business offers its stakeholders; e.g., profits, public safety, entertainment, or convenience
- The characteristics of the system which the SE activities support; e.g., the size, complexity, primary design factors, major components, required products, critical specialties, or areas of life cycle
- The phases of the life cycle in which the SE activities are being performed; e.g., development, deployment, operations, or maintenance of a product or service
- The scale of the business, the systems and services of interest; e.g., is it a single site company or a global venture? Is the business creating a relatively modest product for internal use, such as a new Web application to track employee training, or a new hybrid automobile complete with concerns for engineering, manufacturing, servicing, and distribution?
- The culture of the business in which the SE activities are performed; e.g., is the business risk-averse? Do people normally collaborate or work in isolated organizations?
- The business structure and how well the current structure aligns with what is needed to create new products and services; e.g., does the structure of the business align with the architecture of its major products and services?
- The degree of change or transformation that the business is undertaking in its operation, products, and markets

Rouse (2006) offers a thorough look at enterprise strategy, especially as it relates to delivering value to the enterprise in various phases of the life cycle, beginning with research and development through operations. Rouse provides a number of techniques to determine and improve the value offered to enterprises using SE methods, especially useful when an enterprise is undergoing significant transformation rather than conducting "business as usual"; e.g., the enterprise could be trying to

- do current business better (drive down costs or improve quality of its current products and services);
- cope with a disruption in the market, a competitive threat, or changing customer expectations and ways of doing business;

- reposition itself in its value chain (move from being a part supplier to a subassembly supplier); or
- launch a new generation product or enter a new market.

Eisner (2008) provides a thorough look at different SE organizational approaches.

Systems Engineering Strategy Elements

Based on the primary considerations, the SE strategy generally addresses the following:

- How SE activities provide value to the business (See Economic Value of Systems Engineering)
- How SE activities are allocated among the various business entities (See Organizing Business and Enterprises to Perform Systems Engineering)
- What competencies are expected from the parts of the business in order to perform these SE activities (See Deciding on Desired Systems Engineering Capabilities within Businesses and Enterprises)
- How parts of the business gain and improve competencies (See Developing Systems Engineering Capabilities within Businesses and Enterprises)
- Who performs SE activities within each part of the business (See Team Capability)
- How people who perform SE activities interact with others in the business ((See Part 6: Related Disciplines)
- How SE activities enable the business to address transformation (See Enterprise Systems Engineering).

Depending on the business' approach to SE, there may not be a single coherent SE strategy common across the business. Different business units may have their own SE strategies, or development of a strategy may be delegated to individual projects. The SE strategy may not even be explicitly documented or may only be found in multiple documents across the business. Some businesses publish guidebooks and policies that describe their organizational strategy. These are usually proprietary unless the business is a government or quasi-government agency. Two public documents are NASA (2007) and MITRE (2012). The latter has a number of short articles on different topics including an article on Stakeholder Assessment and Management and another on Formulation of Organizational Transformation Strategies.

Product and Service Development Models

There are three basic product and service development models that most businesses employ:

1. Market-driven commercial
2. Product-line
3. Contract

The biggest differences between the three business models are where requirements risks lie and how user needs and usage are fed into the design and delivery process. SE support to the business varies in each case.

Market-driven commercial products and services are sold to many customers and are typically developed by organizations at their own risk. The requirements come from marketing based on understanding the market, relevant regulation and legislation, and good ideas from within the organization (Pugh 1991, Smith and Reinertsen 1997). Sillitto (1999) contends that market-driven commercial product development is a form of systems engineering with adapted techniques for requirements elicitation and validation.

Product-line products and services are variants of the same product and service, usually customized for each customer. Extra investment is required to create the underlying product platform. Architecting such a platform in a way that supports cost-effective customization is usually more complex both technically and organizationally than market-driven commercial products and services.

Systems engineers typically play a central role in establishing the platform architecture, understanding the implications of platform choices on manufacturing and service, etc. There are a number of examples of good practices in product line; e.g., automobile models from virtually all major manufacturers such as Toyota, General Motors, and Hyundai; Boeing and Airbus aircraft such as the B-737 family and the Airbus 320 family; and Nokia

and Motorola cellphones. The Software Engineering Institute has done extensive research on product lines for software systems and has developed a framework for constructing and analyzing them (Northrop et.al. 2007). For a reference on product line principles and methods, see Simpson (et al. 2006).

Contract products and services often demand tailor-made system/service solutions which are typically specified by a single customer to whom the solution is provided. The supplier responds with proposed solutions. This style of development is common in defense, space, transport, energy, and civil infrastructure. Customers that acquire many systems often have a specific procurement organization with precise rules and controls on the acquisition process, and mandated technical and process standards. The supplier typically has much less flexibility in SE process, tools, and practices in this model than the other two.

Any single business or enterprise is likely to apply some combination of these three models with varying importance given to one or more of them.

Organizations That Use and Provide SE

There are five basic types of organizations that use SE or provide SE services:

1. A business with multiple project teams
2. A project that spans multiple businesses
3. An SE team within either of the above
4. A business with a single project team
5. An SE service supplier that offers a specific SE capability or service (tools, training, lifecycle process) to multiple clients, either as an external consultancy or as an internal SE function.

The kind of business determines the scope and diversity of SE across the organization. This is shown in abstract form in Figure 1, which illustrates the fundamental form of an extended enterprise. This also shows how organizational structure tends to match system structure.

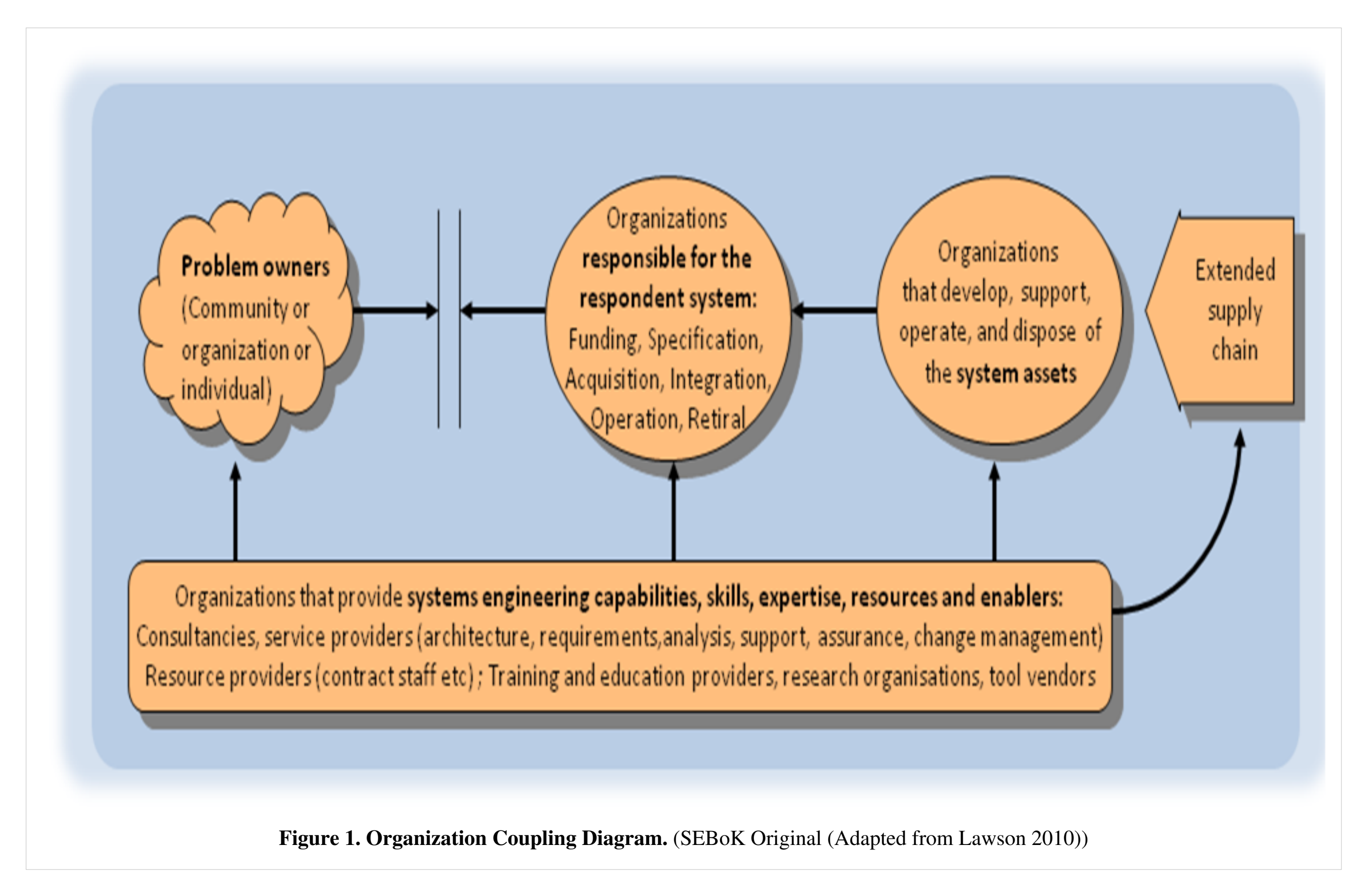


Figure 1. Organization Coupling Diagram. (SEBoK Original (Adapted from Lawson 2010))

The *problem owners* are the people, communities, or organizations involved in and affected by the *problem situation*. They may be seeking to defend a country, to improve transportation links in a community, or to deal with an environmental challenge. The *respondent system* might be a new fighter aircraft, a new or improved transportation infrastructure, or a new low-emission electricity generation systems (respectively). The organizations responsible for the respondent systems would be the Air Force, transport operator or regulator, or electricity supply company. The prime role of these organizations would be to operate the systems of interest to deliver value to the problem owners. They might reasonably be expected to manage the entire system lifecycle.

This same concept is expanded in Figure 2.

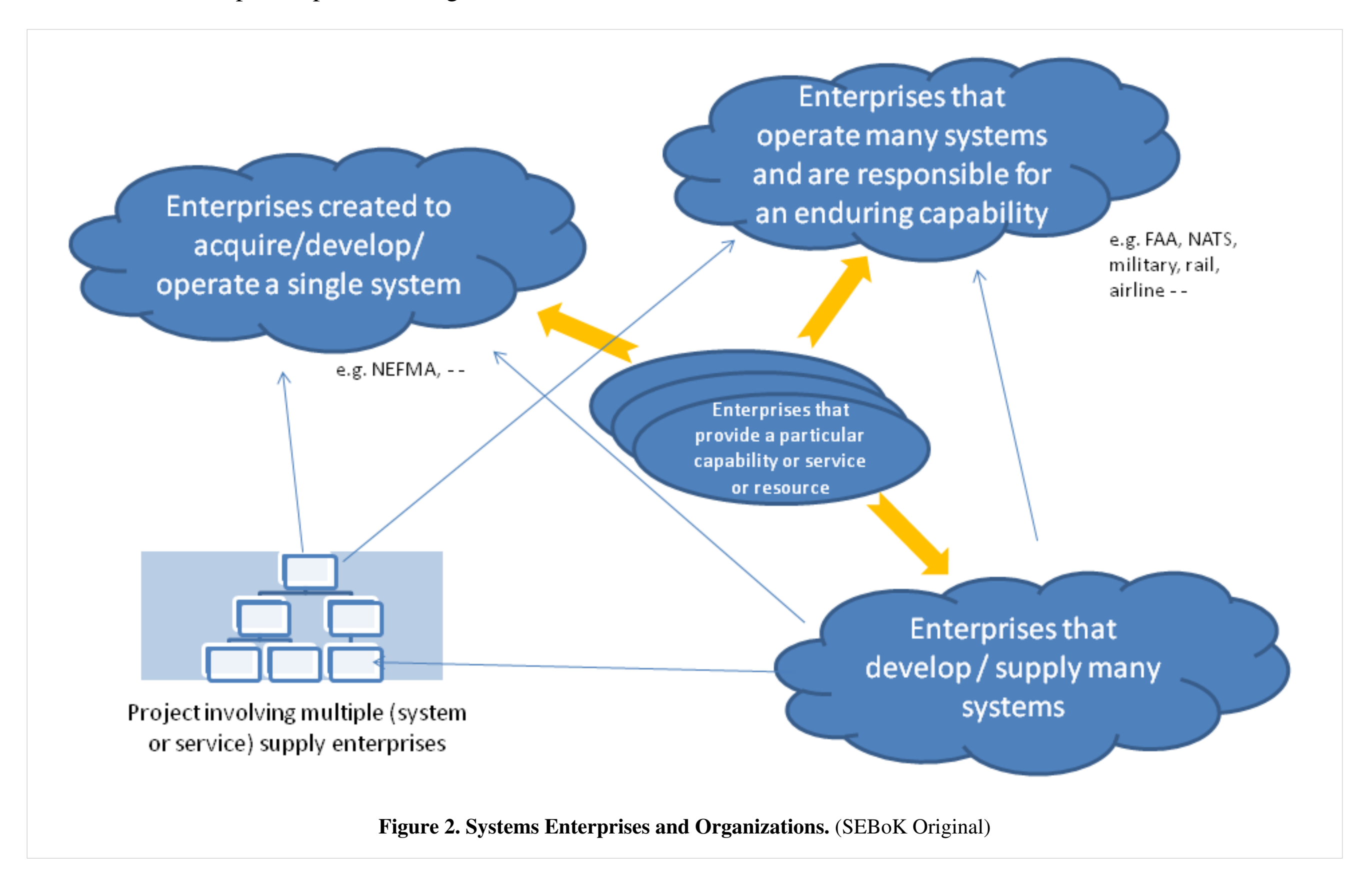


Figure 2. Systems Enterprises and Organizations. (SEBoK Original)

Goals, Measures, and Alignment in a Business

The alignment of goals and measures within the business strongly affects the effectiveness of SE and the benefit delivered by SE to the business, and needs to be carefully understood:

- Blockley and Godfrey (2000) describe techniques used successfully to deliver a major infrastructure contract on time and within budget, in an industry normally plagued by adversarial behavior.
- Lean thinking provides a powerful technique for aligning purpose to customer value – provided the enterprise boundary is chosen correctly and considers the whole value stream (Womack and Jones 2003; Oppenheim et al. 2010).
- Fasser and Brettner (2002, 18-19) see an organization as a system, and advocate three principles for organizational design: (1) increasing value for the ultimate customer, (2) strict discipline, and (3) simplicity.
- EIA 632 (ANSI/EIA 2003) advocates managing all the aspects required for the life cycle success of each element of the system as an integrated "building block". Similarly, Blockley (2010) suggests that taking a holistic view of *"a system as a process"* allows a more coherent and more successful approach to organization and system design, considering each element both as part of a bigger system-of-interest and as a "whole system" (a "holon") in its own right.

- Elliott et al. (2007) advocate six guiding principles for making systems that work: (1) debate, define, revise and pursue the purpose, (2) think holistic, (3) follow a systematic procedure, (4) be creative, (5) take account of the people, and (6) manage the project and the relationships.
- For organizations new to SE, the INCOSE UK Chapter has published a range of one-page guides on the subject, including Farncombe and Woodcock (2009a; 2009b).

Governance

SE governance is the process and practice through which a business puts in place the decision rights that enable SE to deliver as much business value as possible. Those rights may be codified in policy, implemented through the business structure, enforced through tools, and understood through measures of compliance and effectiveness.

SE governance in large businesses is often explicit and codified in policy. In small businesses, it is often tacit and simply understood in how the business works. One of the key implementation steps when a business defines its SE strategy is to establish its SE governance model, which should be tailored to the particular context in which the business operates and delivers value. Of course, in practice, this is often incremental and uneven, and subject to wide swings based on the current state of the business and the people occupying key management positions.

The term governance for development organizations was first popularized in reference to how Information Technology (IT) is overseen in businesses and enterprises (Weill and Ross 2006; Cantor and Sanders 2007). The recognition in the 1990s and the last decade that IT is a fundamental driver of performance and value for most corporations and government agencies led to the transformation of the Chief Information Officer (CIO) into a key senior manager.

Explicit governance of IT became important to enabling an enterprise to respond to new technology opportunities, emerging markets, new threats, and rapid delivery of new products and services. The term "governance" is now widely used to describe how SE is woven into an enterprise. Governance becomes especially challenging for complex projects in which there are high levels of uncertainty (Cantor 2006) or for system of systems projects in which responsibility for major decisions may be distributed over multiple organizations within an enterprise in which there is no single individual who is "in control" (see Systems of Systems (SoS)). Morgan and Liker (2006) describe the governance model for Toyota, which is one of the largest companies in the world.

SE governance establishes the framework and responsibility for managing issues such as design authority, funding and approvals, project initiation and termination, as well as the legal and regulatory framework in which the system will be developed and will operate. Governance includes the rationale and rules for why and how the enterprise policies, processes, methods and tools are tailored to the context. SE governance may also specify product and process measures, documentation standards, and technical reviews and audits.

The ways in which a team organizes to conduct SE activities either conform to policies established at the level above or are captured in that team’s own governance policies, processes, and practices. These policies cover the organizational context and goals, the responsibilities for governance, process, practices and product at the level of interest, and the freedom delegated to and governance and reporting obligations imposed on lower organizational levels. It is good practice to capture the assignment of people and their roles and responsibilities in the form of the Responsible, Accountable, Consult, Inform (RACI) matrix (PMI 2013) or something similar. Responsibility in large organizations can easily become diffused. Sommerville et. al. (2009, 515-529) discuss the relationship between information and responsibility, and describe methods to analyze and model responsibility in complex organizations.

Small organizations tend to have relatively informal governance documentation and processes, while larger organizations tend towards more structure and rigor in their governance approach. Government organizations responsible for developing or acquiring large complex systems, such as the US Department of Defense or the US Federal Aviation Administration, usually develop policies that describe governance of their SE activities and SE organizations. See DoD (2012) for the Department of Defense SE policies.

Government contracting typically brings additional regulation and oversight, driving a group to greater rigor, documentation, and specific practices in their SE governance. Development of systems or operating services that affect public safety or security is subject to constraints similar to those seen in government contracting. Think of the creation of medical devices or the operation of emergency response systems, air traffic management, or the nuclear industry. (See Jackson (2010) for example).

Governance models vary widely. For example, Linux, the greatest success of the open source community, has a governance model that is dramatically different than those of traditional businesses. Smith (2009) offers a cogent explanation of how decisions are made on what goes into the Linux kernel. All of the decision rights are completely transparent, posted on the Linux website, and have proven remarkably effective as they have evolved. The classic paper *The Cathedral and The Bazaar* by Eric Raymond (2000) provides great insight into the evolution of Linux governance and how Linus Torvalds responded to changing context and circumstances to keep Linux so successful in the marketplace with a governance model that was radically novel for its time.

The project management literature also contributes to the understanding of SE governance (see Systems Engineering and Project Management). For example, Shenhar and Dvir (2007) offer the "diamond model" for project management, which identifies four dimensions that should guide how development projects are managed: novelty, technology, complexity, and pace. Application of this model to SE governance would influence the available life cycle models for development projects and how those models are applied.

There are numerous examples of projects that went well or badly based in large part on the governance practiced by both the acquirer and the supplier organizations. Part 7 of the SEBoK has several examples, notably Singapore Water Management (went well) and FAA Advanced Automation System (AAS) (went not so well).

References

Works Cited

ANSI/EIA. 2003. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA). ANSI/EIA 632-1998.

Blockley, D. 2010. "The Importance of Being Process." *Journal of Civil Engineering and Environmental Systems*. 27(3).

Blockley, D. and Godfrey, P. 2000. *Doing It Differently – Systems for Rethinking Construction*. London, UK: Thomas Telford, Ltd.

Cantor, M. 2006. "Estimation Variance and Governance." In IBM developerWorks. Abstract [1] accessed on April 24, 2013.

Cantor, M. and J.D. Sanders. 2007. "Operational IT Governance." In IBM developerWorks. Accessed on September 15, 2011. Available at http://www.ibm.com/developerworks/rational/library/may07/cantor_sanders/.

DoD. 2012. "Systems Engineering Policy". Accessed on August 4, 2012. Available at http://www.acq.osd.mil/se/pg/index.html.

Eisner, H. 2008. "Essentials of Project and Systems Engineering Management", 3rd ed. Hoboken, NJ, USA: John Wiley & Sons.

Elliott, C. et al. 2007. *Creating Systems That Work – Principles of Engineering Systems for The 21st Century.* London, UK: Royal Academy of Engineering. Accessed September 2, 2011. Available at http://www.raeng.org.uk/education/vps/pdf/RAE_Systems_Report.pdf.

Fasser, Y. and D. Brettner. 2002. *Management for Quality in High-Technology Enterprises*. Hoboken, NJ, USA: John Wiley & Sons-Interscience.

Farncombe, A. and H. Woodcock. 2009a. "Enabling Systems Engineering". *Z-2 Guide,* Issue 2.0. Somerset, UK: INCOSE UK Chapter. March, 2009. Accessed September 2, 2011. Available at http://www.incoseonline.org.uk/Documents/zGuides/Z2_Enabling_SE.pdf.

Farncombe, A. and H. Woodcock. 2009b. "Why Invest in Systems Engineering". *Z-3 Guide,* Issue 3.0. Somerset, UK: INCOSE UK Chapter. March 2009. Accessed September 2, 2011. Available at http://www.incoseonline.org.uk/Documents/zGuides/Z3_Why_invest_in_SE.pdf.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions.* Hoboken, NJ, USA: John Wiley & Sons.

Lawson, H. 2010. "A Journey Through the Systems Landscape". London, UK: College Publications, Kings College, UK.

MITRE. 2012. "Systems Engineering Guidebook". Accessed on August 4, 2012. Available at http://www.mitre.org/work/systems_engineering/guide/index.html.

Morgan, J. and J. Liker. 2006. *The Toyota Product Development System: Integrating People, Process and Technology.* New York, NY, USA: Productivity Press.

NASA. 2007. "NASA Systems Engineering Handbook". Accessed on April 24, 2013. Available at http://www.acq.osd.mil/se/docs/NASA-SP-2007-6105-Rev-1-Final-31Dec2007.pdf.Washington, DC, USA: NASA.

Northrop, L., P. Clements, et. al. 2007. *A Framework for Software Product Line Practice*, Version 5.0. With F. Bachmann, J. Bergey, G. Chastek, S. Cohen, P. Donohoe, L. Jones, R. Krut, R. Little, J. McGregor, and L. O'Brien. Pittsburgh, PA, USA: Software Engineering Institute.

Oppenheim, B., E.M. Murman, D.A. Secor. 2010. Lean Enablers for Systems Engineering. *Systems Engineering*. 14(1): 29-55.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Raymond, E.S. 2000. *The Cathedral and The Bazaar,* version 3.0. Accessed on April 24, 2013. Available at http://www.catb.org/esr/writings/homesteading/cathedral-bazaar/cathedral-bazaar.ps.

Rouse, W. 2006. "Enterprise Transformation: Understanding and Enabling Fundamental Change." Hoboken, NJ, USA: John Wiley & Sons.

Shenhar, A.J. and D. Dvir. 2007. *Reinventing Project Management: The Diamond Approach to Successful Growth and Innovation*. Boston, MA, USA: Harvard Business School Publishing.

Sillitto, H. 1999. “Simple Simon Met A System”. Proceedings of the 9th Annual International Council on Systems Engineering (INCOSE) International Symposium, 6-10 June, 1999, Brighton, UK.

Simpson, T.W., Z. Siddique, R.J. Jiao (eds.). 2006. *Product Platform and Product Family Design: Methods and Applications*. New York, NY, USA: Springer Science & Business Media, Inc.

Smith, J.T. 2009. "2.4 Kernel: How are Decisions Made on What Goes into The Kernel?" Available at http://www.linux.com/feature/8090.

Smith, P.G. and D.G. Reinertsen. 1997. *Developing Products in Half the Time*. New York, NY, USA: Wiley and Sons.

Sommerville, I., R. Lock, T. Storer, and J.E. Dobson. 2009. "Deriving Information Requirements from Responsibility Models." Paper presented at 21st International Conference on Advanced Information Systems Engineering, Amsterdam, Netherlands.

Weill, P. and J.W. Ross. 2004. *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*. Boston, MA, USA: Harvard Business School Publishing.Pugh, S. 1991. *Total Design: Integrated Methods for Successful Product Engineering.* New York, NY, USA: Addison-Wesley.

Womack, J. and D. Jones. 2003. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*, Revised Edition. New York, NY, USA: Simon & Schuster.

Primary References

Blockley, D. and Godfrey, P. 2000. *Doing It Differently – Systems for Rethinking Construction*. London, UK: Thomas Telford, Ltd.

Cantor, M. and J.D. Sanders. 2007. "Operational IT Governance." In IBM developerWorks. Accessed on September 15, 2011. Available at http://www.ibm.com/developerworks/rational/library/may07/cantor_sanders/.

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons.

Elliott, C., et al. 2007. *Creating Systems That Work – Principles of Engineering Systems for The 21st Century.* London, UK: Royal Academy of Engineering. Accessed September 2, 2011. Available at http://www.raeng.org.uk/education/vps/pdf/RAE_Systems_Report.pdf.

Lawson, H. 2010. *A Journey Through the Systems Landscape*. London, UK: College Publications, Kings College, UK.

Morgan, J. and J. Liker. 2006. *The Toyota Product Development System: Integrating People, Process and Technology*. New York, NY, USA: Productivity Press.

Northrop, L., P. Clements, et al. 2007. *A Framework for Software Product Line Practice,* version 5.0. With F. Bachmann, J. Bergey, G. Chastek, S. Cohen, P. Donohoe, L. Jones, R. Krut, R. Little, J. McGregor, and L. O'Brien. Pittsburgh, PA, USA: Software Engineering Institute. Accessed on April 25, 2013. Available at http://www.sei.cmu.edu/productlines/frame_report/index.html.

Rouse, W. 2006. *Enterprise Transformation: Understanding and Enabling Fundamental Change.* Hoboken, NJ, USA: John Wiley & Sons.

Shenhar, A.J. and D. Dvir. 2007. *Reinventing Project Management: The Diamond Approach to Successful Growth and Innovation*. Boston, MA, USA: Harvard Business School Publishing.

Additional References

Chastek, D., P. Donohoe, and J.D. Mcgregor. 2009. *Formulation of a Production Strategy for a Software Product Line*. Pittsburgh, PA, USA: Software Engineering Institute, CMU/SEI-2009-TN-025. Accessed on September 14, 2011. Available at http://www.sei.cmu.edu/reports/09tn025.pdf.

Sillitto, Mazzella, and Fromenteau. 2001. "The development of Product Lines in THALES: methods, examples, lessons learnt," Paper presented at the INCOSE UK Spring Conference.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.ibm.com/developerworks/library/?sort_by=&show_abstract=true&show_all=&search_flag=&contentarea_by=All+Zones&search_by=Estimation+Variance+and+Governance&product_by=-1&topic_by=-1&industry_by=-1&type_by=All+Types&ibm-search=Search

Determining Needed Systems Engineering Capabilities in Businesses and Enterprises

Enabling a business or enterprise to perform systems engineering (SE) well requires deciding which specific SE capabilities the business or enterprise needs in order to be successful. (In the rest of this article business or enterprise is usually abbreviated to just "business", because a business is a specific type of enterprise that has sufficiently strong central authority and motivation to take steps to enable SE). SE capabilities should support the Systems Engineering Organizational Strategy and reflect the nature of the business, its products and services, various stakeholders, business leadership focus, etc.

This topic, which is part of the Enabling Businesses and Enterprises knowledge area (KA) of Part 5, summarizes the factors used to decide which SE capabilities a business needs; e.g., the interactions between SE and other functional areas in the business, and consideration of social dynamics and leadership at the team and business levels. Needed capabilities may be decided and developed centrally by a business, or within teams and individuals, or through some combination of the two. Determination of team SE capability is discussed in the article Team Capability, and individual SE competencies are discussed in the article Roles and Competencies.

Relationship of this Topic to Enterprise Systems Engineering

Enterprise Systems Engineering and Capability Engineering techniques can be used to establish needed SE capabilities. At a high level of abstraction, the following are basic steps that could be used to decide the desired SE capabilities within the business:

1. understand the context;
2. determine the required SE roles;
3. determine the competencies and capabilities needed for each of the SE roles;
4. assess the ability and availability of the needed SE organizations, teams, and individuals;
5. make adjustments to the required SE roles based on the actual ability and availability; and
6. organize the SE function to facilitate communication, coordination, and performance.

See the article Organizing Business and Enterprises to Perform Systems Engineering for additional information. More information on context and required SE roles is provided below.

Contextual Drivers

The following discussion illustrates some of the contextual factors that influence the definition of the SE capability needed by a business.

Where the SE Activities are Performed in the Value Chain

The SE approach adopted by the business should depend on what role the organization plays. Ring (2002) defines a value cycle, and where the business sits in that cycle is a key influence of SE capability need.

- **Problem owner:** focus on identifying and scoping the system problem (defining system-of-interest (SoI)(glossary))and understanding the nature of the appropriate respondent system using Enterprise Systems Engineering and Capability Engineering approaches.
- **System operator:** focus on establishing all the necessary components of capability (glossary) to deliver the required services, as well as on integrating new system assets into the system operation as they become available (see Service Systems Engineering). The definition of the components of capability varies by organization - e.g.,
  - The US Department of Defense defines the components of capability as DOTMLPF: doctrine, organization, training, materiel, logistics, people, and facilities.

    - The UK Ministry of Defense defines the components of capability as TEPIDOIL; i.e., training, equipment, people, information, doctrine, organization, infrastructure, and logistics.
    - Other domains and organizations define the components of capability with similar, equivalent breakdowns which are either explicit or implicit.
- **Prime contractor or primary commercial developer:** focus on understanding customer needs and trading alternative solution approaches, then establishing a system team and supply chain to develop, deliver, support, and in some cases, operate the system solution. This may require enterprise SE (see Enterprise Systems Engineering) as well as "traditional" product SE (see Product Systems Engineering).
- **Subsystem/component developer:** focus on understanding the critical customer and system integrator issues for the subsystem or component of interest, define the component or subsystem boundary, and integrate critical technologies. This may exploit re-usable elements and can be sold in identical or modified forms to several customers. (In Part 4 of the SEBoK, see Systems of Systems, Enterprise Systems Engineering, and Product Systems Engineering for more information and references to the literature.)
- **Specialist service provider:** focus on specific process capabilities and competences which are typically sold on a time and materials or work package basis to other businesses.

Where the Enterprise Operates in the Lifecycle

The SE capabilities required by the business will depend on the system life cycle (glossary) phase(s) in which it operates (see Life Cycle Models in Part 3).

- **Concept definition phase:** requires the SE capability to identify a “problem situation,” define the context and potential concept of operations for a solution system, assess the feasibility of a range of possible solutions in broad terms, and refine the definition to allow the development of system requirements for the solution (see Concept Definition in Part 3).
- **System Definition phase:** requires the SE capability to influence concept studies (ensure feasible and understood by the development team), establish the trade space that remains at the end of the concept study, perform the system definition activities, including architecture design, and create a detailed definition of the system elements.
- **System realization phase:** requires the SE capability to configure the manufacturing and logistics systems for the system assets, and manufacture system assets (see System Realization in Part 3).
- **System deployment and use:** requires the SE capability to maintain business continuity during the transition to operation, bring the system into service, support system, monitor system performance, and respond to emerging needs (see System Deployment and Use. Elliott et al. (2008) describe the different emphases that should be placed in SE during the "in-service" phase. This phase particularly requires the business to be able to perform SE at an appropriate operational tempo.
- **Retirement phase:** requires the SE capability for ensuring the safe retirement of systems and keeping them in a state ready for re-activation (“mothballed”), safe disposal of the system assets.

Nature of Responsibility to End Users and Society

Depending on the business model and the contracting environment, the business may find that its responsibility to end users is

- **explicit**, or spelled out by clear requirements and prescriptive legislation; or
- **implicit**; i.e., a legal or ethical obligation to ensure “fitness for purpose” which may be enforced by commercial frameworks, national or international standards, and specific product liability legislation.

Typically, businesses whose business model is contract driven focus on satisfying explicit requirements, whereas market-driven businesses have to be more aware of implicit responsibilities.

Nature of Responsibility to Customers

The business may contract with its customers to deliver any of the following:

- **an outcome**: The intended benefits the system is expected to provide, requires enterprise systems engineering;
- **an output**: Deliver or operate the system or part of it against agreed acceptance criteria; requires product systems engineering;
- **an activity**: Perform a specified set of tasks, requires service systems engineering; and
- **a resource**: Provide a specified resource; requires focus on individual competencies - see Enabling Individuals.

Scale of Systems

The business or enterprise may need very different SE approaches depending on the scale of the system at which the business operates. The following categories are based on Hitchins' five layered system model (Hitchins 2005):

- **Level 1: Subsystem and technical artifacts** – focus on product systems engineering and on technology integration.
- **Level 2: Project systems** – focus on product systems engineering with cross-discipline and human integration.
- **Level 3: Business systems** – focus on enterprise systems engineering , service systems engineering to implement them, and on service management (Chang 2010) and continuous improvement (SEI 2010b); see also Quality Management) for the day to day running of the business.
- **Level 4: Industry systems** – If there is a conscious effort to treat an entire industry as a system, the focus will be on Enterprise Systems Engineering, and on the long-term economic and environmental sustainability of the overall industry.
- **Level 5: Societal systems** – Enterprise systems engineering is used to analyze and attempt to optimize societal systems (see Singapore Water Management in Part 7).

Sillitto (2011) has proposed extending this model to cover sustainability issues by adding two additional layers, the "ecosystem" and the "geosystem".

Complexity of Systems Integration Tasks and Stupples' levels

Creating Systems That Work – Principles of Engineering Systems for The 21st century identifies three "kinds" of SE, originally proposed by Stupples (2006), that have to do with the level of cross-disciplinary integration involved (Elliot et al. 2007)

1. Within a discipline (e.g., software, hardware, optics, *or* mechanics), the SE focus is on taking a systems view of the architecture and implementation to manage complexity and scale within a single engineering discipline.
2. In multiple disciplines (e.g., software, hardware, optics, *and* mechanics), the SE focus is on holistic integration of multiple technologies and skills to achieve a balanced system solution.
3. In socio-technical systems integration, the SE focus is on getting people and the non-human parts of the system working synergistically.

Sillitto (2011) proposed extending this model properly to cover sustainability issues by adding one additional level, "Environmental Integration". He describes this level and show how the Stupples' levels relate to other dimensions used to categorize systems and professional engineering skills.

Criticality of System and Certification Requirements

The level of rigor in the SE approach adopted by the business will depend on the criticality of various classes of requirement. (See Systems Engineering and Specialty Engineering.)

- Safety and security requirements often demand specific auditable processes and proof of staff competence.
- Ethical and environmental requirements may require an audit of the whole supply and value chain.
- Extremely demanding combinations of performance requirements will require more design iteration and more critical control of component characteristics; e.g., see Quality Management and *Management for Quality in High-Technology Enterprises* (Fasser and Brettner 2010).

The Nature of a Contract or Agreement

The nature of the contractual relationship between a business and its customers and end users will influence the style of SE.

- Fixed price, cost plus, or other contracting models influence the mix of focus on performance and cost control and how the business is incentivized to handle risk and opportunity.
- In mandated work share arrangements, the architecture of the product system may be compromised or constrained by the architecture of a viable business system; this is often the case in multi-national projects and high profile government procurements (Maier and Rechtin 2009, 361-373).
- In self-funded approaches, the priorities will be requirements elicitation approaches designed to discover the latent needs of consumers and business customers, as well as development approaches designed to achieve rapid time to market with a competitive offering, or to have a competitive offering of sufficient maturity available at the most critical time during a customer's selection process.
- In single phase or whole-life approaches, the business may be able to optimize trade-offs across the development, implementation, and in-service budgets, and between the different components of capability (glossary).

The Nature and Predictability of Problem Domain(s)

Well-defined and slowly-changing technologies, products, and services permit the use of traditional SE life cycle models based on the waterfall model because the requirements risk and change is expected to be low (see Life Cycle Models).

Poorly defined and rapidly changing problem domains, with operators subject to unpredictable and evolving threats, demand more flexible solutions and agile processes. SE should focus on modular architectures that allow rapid reconfiguration of systems and systems-of-systems, as well as rapid deployment of new technologies at a subsystem level to meet new demands and threats.

Fundamental Risks and Design Drivers in the Solution Domain

When the solution domain is stable, with a low rate of technology evolution, and systems use mature technology, the focus is on optimum packaging and configuration of known and usually well-proven building blocks within known reference architectures, and on low-risk incremental improvement over time.

When there is rapid technology evolution, with pressure to bring new technologies rapidly to market and/or into operational use, the SE approach has to focus on technology maturation, proof of technology and integration readiness, and handling the technology risk in the transition from the lab to the proof of concept to the operational system.

There is usually a trade-off between lead time expectations and the level of integrity/certification. In the development of new systems, short lead times are seldom compatible with high levels of system integrity and rigorous certification.

Competitive Situation and Business Goals

The business drivers for SE deployment may be one or more of the following:

- To perform existing business better;
- To recover from a competitive shock or a shift in clients' expectations;
- To develop a new generation product or service;
- To enter a new market; and/or
- To reposition the business or enterprise in the value chain.

In the first case, SE can be deployed incrementally in parts of the business process where early tangible benefits can be realized. This could be the early steps of a business-wide strategic plan for SE. (See Systems Engineering Organizational Strategy for more on setting SE strategy and Developing Systems Engineering Capabilities within Businesses and Enterprises for improving SE capabilities.)

In the other cases, the business is going through disruptive change and the early priority may be to use systems thinking (see Systems Thinking) and enterprise SE approaches to scope the transformation in the context of a major change initiative.

Type of System or Service

There are three distinct flavors of products or service types (see Systems Engineering Organizational Strategy):

1. In a product or productized service, the focus will be on predicting how the market might change during the development period, eliciting, anticipating, and balancing requirements from a variety of potential customers, and optimizing features and product attractiveness against cost and reliability.
2. In a custom solution (product or service) the focus will be on feasible and low-risk (usually) approaches to meet the stated requirement within budget, using system elements and technologies that are known or expected to be available within the desired development timescale.
3. Tailored solutions based on standard product and/or service elements require a much more sophisticated SE process that is able to use a “product line approach” to blend standard modules with planned adaptation to meet clients’ specific needs more quickly and cheaply than would be possible with a single contract solution. The business needs to manage the life cycle and configuration of the standard modules separately from, but coherently with, the life cycle and configuration of each tailored solution.

Needed Systems Engineering Roles

After understanding the context for the business, the next step is to determine the SE capabilities required in the role in the business. The SEI Capability Maturity Models for acquisition, development, and services (SEI 2007; SEI 2010a; SEI 2010b) provide a framework for selecting SE capabilities relevant to different types of business. Existing SE competency models can be used to assist in determining the needed capabilities. An example is the INCOSE SE Competencies Framework (INCOSE 2010). (See Roles and Competencies for more information on competency models.).

There can be a wide spectrum of the spread of SE focus, from SE being focused in a specialist role, an interface or glue role (Sheard 1996), or the idea that “SE is good engineering with special areas of emphasis… including interfaces between disciplines” (Blanchard and Fabrycky 2005) and so it is shared by all. In any organization where activities and skills are shared, there is always a danger of silos or duplication.

As part of the role definition, the business must define where an individual doing SE fits into career progression (what roles before SE, what after?). Developing Individuals describes how individuals improve SE; the organization must define the means by which that development can be enacted. Businesses need to customize from a range of development strategies; see, for example, Davidz and Martin (2011).

As shown in Figure 1 below, management action on workforce development will be required if there are systemic mismatches between the competencies required to perform SE roles and the actual competencies of individuals. The organizational culture may have a positive or negative effect on team performance and the overall value added by the business (see Culture).

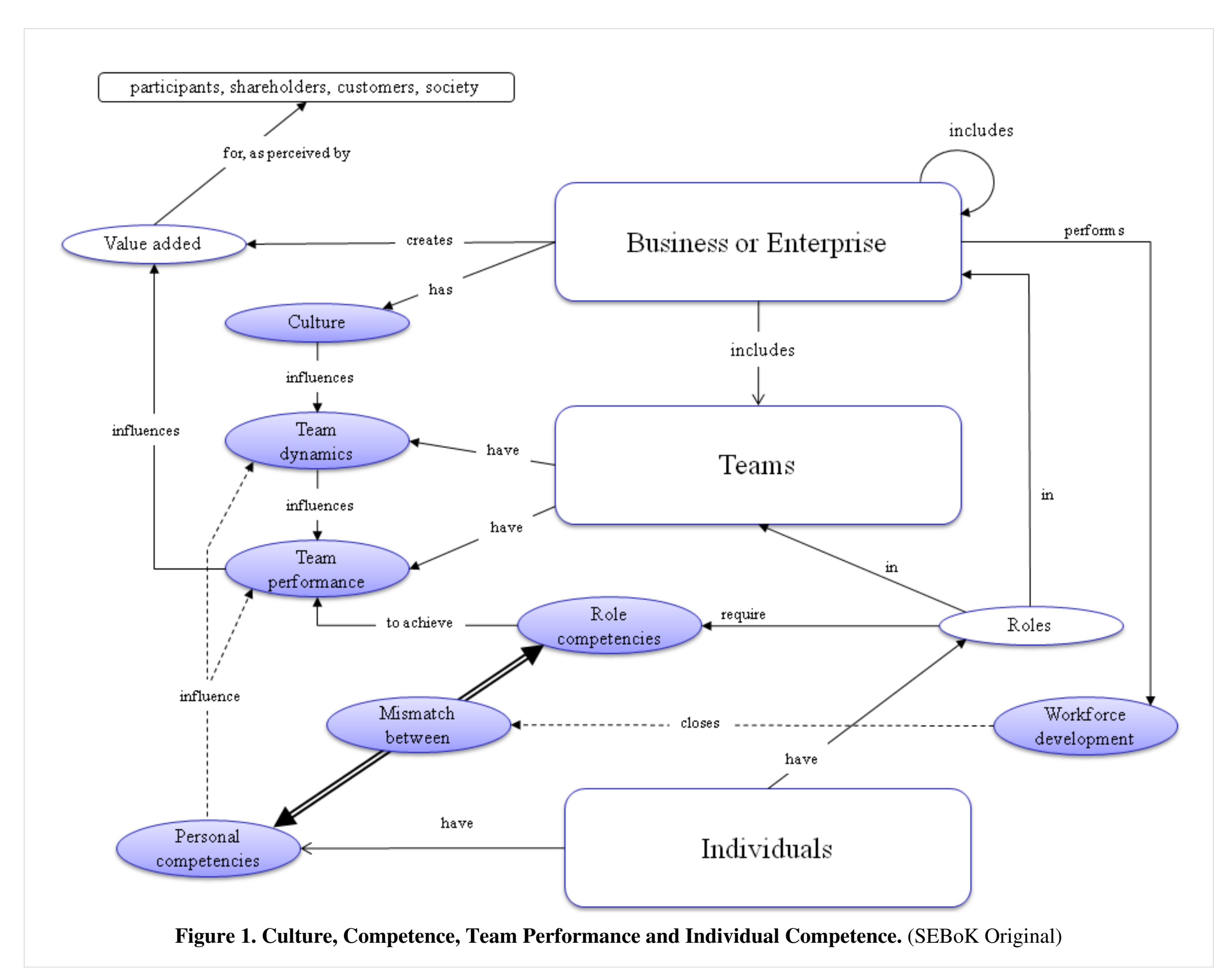


Figure 1. Culture, Competence, Team Performance and Individual Competence. (SEBoK Original)

Required SE Processes and Methods

The decisions on how to implement SE capability must be embedded in the businesses processes and its availability methodologies and toolsets. Embedding SE principles, processes, and methods in the organization's quality management system means that senior management and the quality system will help embed SE in the organizational business process and make sure it is applied (INCOSE 2012; ISO/IEC 2008; see Quality Management).

When defining the processes and tools, a balance between the need for a systematic and standardized approach to SE processes, such as that seen in INCOSE (2012), with the flexibility inherent in systemic thinking is critical. Systems thinking helps the organization understand problem situations, remove organizational barriers, and make the most of the organization's technical capabilities (see Beasley (2011)).

Need for Clarity in the SE Approach and the Dangers of Implementing SE

Clarity on how the organization does SE is important. Typically, implementing SE may be part of an organization's improvement, so Kotter's principles on creating a vision, communicating the vision, and empowering others to act on the vision are extremely relevant (Kotter 1995). The way an organization chooses to do SE should be part of the vision of the organization and must be understood and accepted by all.

Many of the major obstacles in SE deployment are cultural (see Culture).

One of the lean enablers for SE is to "pursue perfection" (Oppenheim et al. 2010). The means of improvement at a business or enterprise level are discussed in detail elsewhere, but the starting point has to be deciding what SE capabilities the organization wants. It needs to be recognized that the needed capabilities change over time (learning, improving, or losing capability). Thus, balancing SE with everything else that it involves is an ever changing process.

References

Works Cited

Beasley, R. 2011. "The Three T's of Systems Engineering." Paper presented at the 21st Annual International Council on Systems Engineering (INCOSE) International Symposium. June 2011. Denver, CO, USA.

Blanchard, B. and W. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th edition. Upper Saddle River, NJ, USA: Prentice Hall.

Chang, C.M. 2010. Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence. Hoboken, NJ, USA: John Wiley and Sons.

Davidz, H. L. and J. Martin. 2011. "Defining a Strategy for Development of Systems Capability in the Workforce." Systems Engineering. 14(2) (Summer, 2011): 141-143

Elliott, B. et al. 2008. *INCOSE UK Chapter Working Group on Applying Systems Engineering to In-Service Systems,* Final Report. Somerset, UK: INCOSE UK Chapter Working Group. Accessed September 6, 2011. Available at http://www.incoseonline.org.uk/Documents/Groups/InServiceSystems/is_tr_001_final_report_final_1_0.pdf.

Fasser, Y. and D. Brettner. 2001. *Management for Quality in High-Technology Enterprises*. New York, NY, USA: Wiley.

Hitchins, D. 2005. *Systems Engineering 5 Layer Model.* Accessed on April 24,2013. Available at http://www.hitchins.net/systems/world-class-systems-enginee.html.

INCOSE. 2010. *SE Competencies Framework*, Issue 3. Somerset, UK: International Council on Systems Engineering (INCOSE), INCOSE Technical Product 2010-0205.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2008.

Kotter, J. 1995. *Leading Change: Why Transformation Efforts Fail*. Boston, MA, USA: Harvard Business Review (March–April 1995).

Maier, M. and E. Rechtin. 2009. *The Art of System Architecting, Third Edition*. Boca Raton, FL, USA: CRC Press.

Oppenheim et al. 2010. *Lean Enablers for Systems Engineering*. New York, NY, USA: Wiley and Sons, Inc.

Ring J. 2002. *Toward an Ontology of Systems Engineering.* INSIGHT, 5(1): 19-22

SEI. 2007. *CMMI for Acquisition*. Version 1.2. Technical Report CMU/SEI-2007-TR-017. Pittsburgh, PA, USA: Software Engineering Institute, Carnegie Mellon University.

SEI. 2010a. *Capability Maturity Model Integrated (CMMI) for Development*. Version 1.3. Pittsburgh, PA, USA: Software Engineering Institute, Carnegie Mellon University.

SEI. 2010b. *CMMI for Services*. Version 1.3. Technical Report CMU/SEI-2010-TR-034. Pittsburgh, PA, USA: Software Engineering Institute, Carnegie Mellon University.

Sheard, S. 1996. "12 Systems Engineering Roles." Paper presented at the 6th Annual International Council on Systems Engineering (INCOSE) International Symposium. Boston, MA, USA. Accessed September 14, 2011.

Sillitto, H. 2011. "Unravelling Systems Engineers from Systems Engineering - Frameworks for Understanding the Extent, Variety and Ambiguity of Systems Engineering and Systems Engineers." Paper presented at the 21st Annual International Council on Systems Engineering (INCOSE) International Symposium. 20-23 June 2011. Denver, CO, USA.

Stupples, D. 2006. "Systems Engineering – a road from perdition." Published on Royal Academy of Engineering website - available at http://www.raeng.org.uk/education/vps/systemdesign/pdf/David_Stupples.pdf

Primary References

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology*. Chichester, UK: Wiley and Sons, Inc.

Oppenheim, B. 2011. *Lean for Systems Engineering - with Lean Enablers for Systems Engineering*. Hoboken, NJ, USA: Wiley and Sons, Inc.

Sheard, S. 1996. *Twelve Systems Engineering Roles.* Paper presented at the 6th Annual International Council on Systems Engineering (INCOSE) International Symposium. Boston, MA, USA. Accessed September 14, 2011.

Additional References

Rhodes, D., and G. Roedler (eds.). 2007. *Systems Engineering Leading Indicators Guide*, version 1.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2005-001-02.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Organizing Business and Enterprises to Perform Systems Engineering

In order for a business or enterprise to perform systems engineering (SE) well, the team must decide which specific SE capabilities the business or enterprise needs in order to be successful and then organizing to deliver those capabilities. (In the rest of this article, business or enterprise is usually abbreviated to just "business", because a business is a specific type of enterprise that has sufficiently strong central authority and motivation to take steps to enable SE).

SE capabilities and organizational approach should support the Systems Engineering Organizational Strategy and reflect the nature of the business, its products and services, various stakeholders, business leadership focus, etc. This topic, which is part of Part 5, Enabling Businesses and Enterprises, summarizes the factors used to organize a business to perform SE.

Components of Business and Enterprise SE Capability

Organization Issues - Culture, Knowledge, Information, and Infrastructure

The way SE is managed is described in Systems Engineering Organizational Strategy, which both impacts and responds to the SE culture and approach.

Knowledge and Information

Knowledge and Information are key assets in a business, and their management is critical. Fasser and Brettner (2002) discuss knowledge management extensively. They assert that *“We may think that knowledge transfer is just an information technology issue, but in actuality, it is also a psychological, cultural, and managerial issue – in short a human issue” and “Only information in action can create knowledge”.*

Organizations need to manage SE know-how, integration of SE with other organizational processes and activities, and knowledge of their business domain. The INCOSE Intelligent Enterprise Working Group's work on knowledge management in an SE context led to the publication of a *“Concept of Operations for a Systems Engineering Educational Community”* (Ring et al. 2004).

Information has to be both shared and protected in complex organizations. Sharing is key to effective collaboration and is constrained by the need to protect intellectual property, as well as commercially and nationally sensitive material. Different cultures and personal styles use information in different ways and in different orders. (Levels of abstraction, big picture first or detail, principles first or practical examples, etc.) Sillitto (2011b) describes the knowledge management challenges for large, multi-national organizations.

Projects need to manage project information and establish configuration control over formal contractual information, as well as the information that defines the product/service being developed, supplied, or operated. A key role of systems engineers is to “language the project” (Ring et al. 2004). Good data management and tool support will allow people to document once, use many times, and ensures consistency of information over time and between different teams.

System information needs to be maintained throughout the life of the system and made available to relevant stakeholders – including those designing new systems that have to interface to the system-of-interest - to allow system management, maintenance, reconfiguration, upgrade and disposal, and forensics after accidents and near-misses. Elliott et al. (2008) suggest that information management is the dominant problem in SE in service systems, and that the cost and difficulty of establishing current state and legacy constraints before starting to implement a change is often underestimated.

"Infostructure" (information infrastructure) to support the system lifecycle will include the following:

- Information assets such as process libraries, document templates, preferred parts lists, component re-use libraries, as-specified and as-tested information about legacy systems, capitalized metrics for organizational performance on previous similar projects, all with appropriate configuration control
- Modeling and simulation tools, data sets and run-time environments
- Shared working environments ─ workspaces for co-located teams, areas for people to interact with each other to develop ideas and explore concepts, work areas suitable for analysis tasks, meeting rooms, access control provision, etc.
- IT facilities - computer file structures, software licenses, IT equipment, computer and wall displays to support collaborative working, printers, all with appropriate security provision and back-up facilities, procedures for efficient use, and acceptable performance and usability
- Security provisions to protect own, customer, supplier and third party IPR and enforce necessary protective working practices while allowing efficient access to information for those with a need to know

SE is a knowledge activity. Systems engineers need appropriate facilities for accessing, sharing and capturing knowledge, as well as for interacting effectively with the whole set of stakeholders. Warfield (2006) describes collaborative workspaces, environments and processes for developing a shared understanding of a problem situation.

Enabling Infrastructure

The ISO/IEC 15288 (ISO 2008) Infrastructure Management Process provides the enabling infrastructure and services to support organization and project objectives throughout the life cycle. Infrastructure to support the system life cycle will often include the following:

- Integration and test environment ─ bench and lab facilities, facilities for development testing as well as acceptance testing at various levels of integration, calibration and configuration management of test environments
- Trials and validation environment ─ access to test ranges, test tracks, calibrated targets, support and storage for trials-equipment, harbor, airfield and road facilities, safe storage for fuel, ordinance, etc.
- Training and support infrastructure ─ training simulators, embedded training, tools and test equipment for operational support and maintenance, etc.

People

The roles people fill are typically defined by the business/enterprise (see Determining Needed Systems Engineering Capabilities in Businesses and Enterprises), although those decisions may be pushed down to teams. Enabling Teams explains how people are used in teams; Enabling Individuals describes the development of an individual's SE competence.

The implementation of these roles needs further consideration. Sheard (1996) lists twelve system engineering roles. Sheard (2000) draws an important distinction between roles involved in the discovery phase, characterized by a high level of uncertainty, the program phase, which is more deterministic and defined, and the overall systems engineering approach. Kasser et al. (2009) identify five types of systems engineer distinguished by the need to work at increasing levels of abstraction, ambiguity, scope and innovation. Sillitto (2011a) discusses a number of SE roles and the characteristics required of them, in the context of the wider engineering and business professional landscape.

Systems engineering exists within an enterprise “ecosystem.” Two key aspects to consider:

- How much should the business/enterprise nurture and value the systems engineer?
- How much should the business/enterprise pull value from systems engineers, rather than wait for systems engineers to "push" value on the business/enterprise?

Process

Many SE organizations maintain a set of organizational standard processes which are integrated in their quality and business management system, adapted to their business, and with tailoring guidelines used to help projects apply the standard processes to their unique circumstances. Guidance on organizational process management is provided by such frameworks as the Capability Maturity Model Integration (CMMI) (SEI 2010), which has two process areas on organizational process: Organizational Process Development (OPD) is concerned with organizational definition and tailoring of the SE lifecycle processes (discussed in detail elsewhere in this document) and Organizational Process Focus (OPF), which is concerned with establishing a process culture in an organization.

To document, assess, and improve SE processes, businesses often establish a systems engineering process group. Members of such groups often create standard process assets, may mentor teams and business units on how to adopt those standard processes and assess how effective those processes are working. There is a large body of literature on SE process improvement based on various process improvement models. Two of the most popular are ISO/IEC 9000 (2000) and CMMI (SEI 2010). The Software Engineering Institute, which created the CMMI, offers many free technical reports and other documents on CMMI at http://www.sei.cmu.edu/cmmi.

Assessment and measuring process performance is covered in Assessing Systems Engineering Performance of Business and Enterprises.

Tools and Methods

SE organizations often invest in SE tools and models, develop their own, and/or integrate off-the-shelf tools into their particular business/enterprise processes. Tools require great attention to culture and training; to developing a consistent “style” of use so that people can understand each others’ work; and proper configuration and management of the information so that people are working on common and correct information.

It is important that methods are used as well as tools, particularly to support Systems Thinking.

It is common practice in large SE organizations to have a tool support infrastructure which ensures that tools support the organizational standard processes and are fully integrated with training, and that projects and teams can use the tools to do their job and are not distracted by tool management issues that are more efficiently handled centrally. Smaller SE organizations often operate more informally.

Fitting It All Together

The concept map in Figure 1 below shows the relationships between the various aspects of organization, resource, responsibility, and governance.

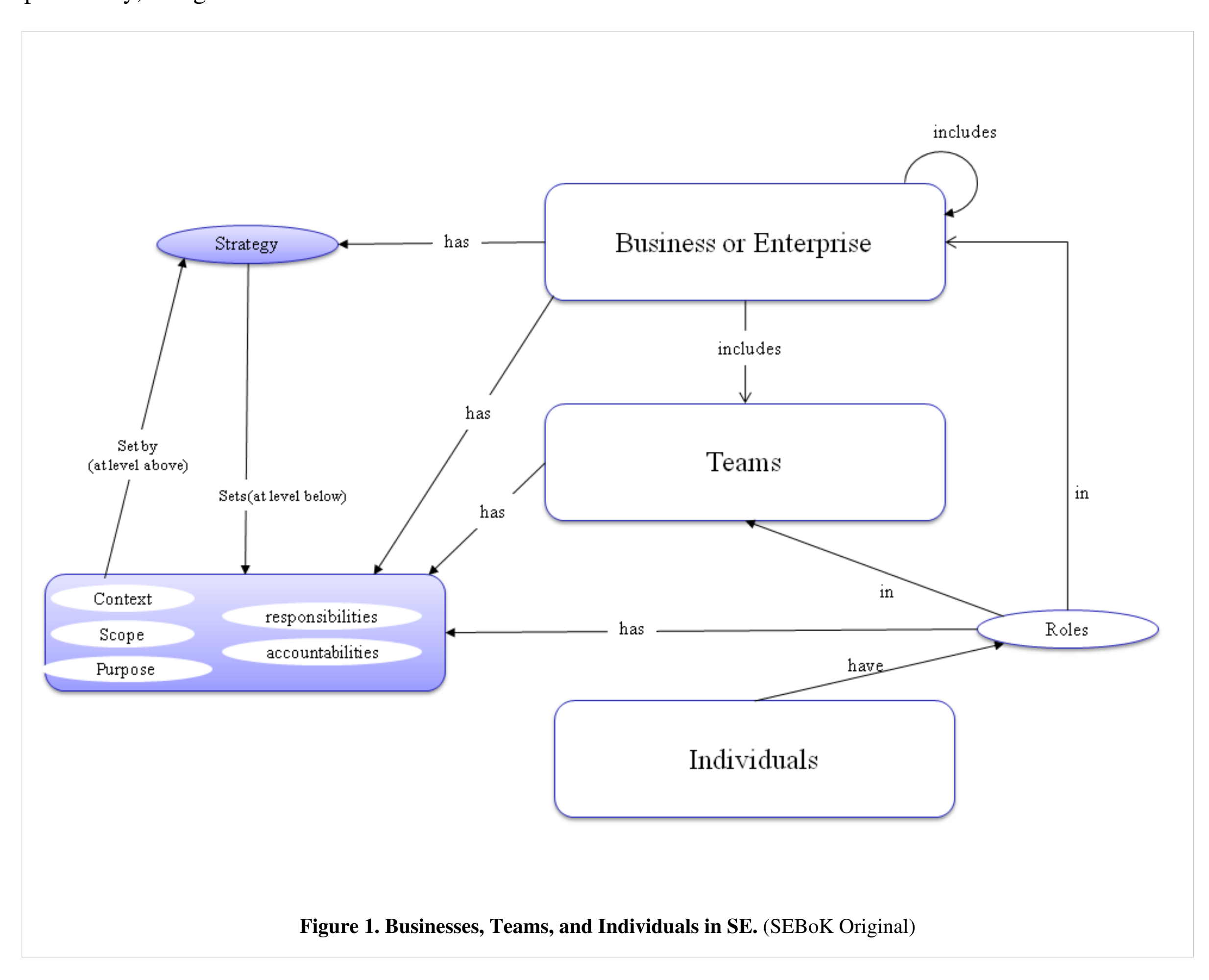


Figure 1. Businesses, Teams, and Individuals in SE. (SEBoK Original)

Enterprise Structures and Their Effects on SE

Enterprises manage SE resources in many different ways. A key driver is the extent to which they seek to optimize use of resources (people, knowledge, and assets) across teams and across the enterprise as a whole. Five common ways of organizing resources to support multiple projects are: project; matrix; functional; integrated; and product centered (CM Guide 2009, Handy 1985, PMI 2013, section 2.1.3). A large enterprise would likely apply some combination of these five ways across its constituent sub-enterprises and teams. Browning (2009) offers a way to optimize project organizational structure. Eisner (2008) offers a good overview of different organizational models.

Project Organization

A project organization is one extreme in which projects are responsible for hiring, training, and terminating staff, as well as managing all assets required for delivery. In this model, systems engineers on a project report to the project manager and resources are optimized for the delivery of the project. This model has the advantage of strongly aligning the authority and responsibility of the project with the project manager. However, it operates at the expense of sub-optimizing how the staff is deployed across the larger enterprise, how technology choices are made across projects, etc. *Systems Engineering Fundamentals* (DAU 2001) offers a DoD view of good practice project organizations.

Functional Organization

A functional organization demonstrates the opposite extreme. In a functional organization projects delegate almost all their work to functional groups, such as the software group, the radar group or the communications group. This is appropriate when the functional skill is fast-evolving and dependent on complex infrastructure. This method is often used for manufacturing, test engineering, software development, financial, purchasing, commercial, and legal functions.

Matrix Organization

A matrix organization is used to give systems engineers a “home” between project assignments. Typically, a SE functional lead is responsible for career development of the systems engineers in the organization, a factor that influences the diversity and length of individual project assignments.

Integrated Organization

In an integrated organization, people do assigned jobs without specific functional allegiance. Those that perform SE tasks are primarily identified as another type of engineer, such as a civil or electrical engineer. They know systems engineering and use it in their daily activities as required.

Product Centered Organization

In accordance with the heuristic (glossary) that “the product and the process must match” (Rechtin 1991, 132), a common method for creating an organizational structure is to make it match the system breakdown structure (SBS) (glossary). According to Browning (2009), at each element of the SBS there is an assigned integrated product team (IPT)(glossary). Each IPT consists of members of the technical disciplines needed to design the product system. The purpose of the IPT is to assure that the interactions among all the technical disciplines are accounted for in the design and that undesirable interactions are avoided.

Interface to Other Organizations

Outside official engineering and SE organizations within an enterprise, there are other organizations whose charter is not technical. Nevertheless, these organizations have an important SE role.

- **Customer Interface Organizations:** These are organizations with titles such as Marketing and Customer Engineering. These are the organizations with the most direct interface with current or potential clientele. Their role is to determine customer needs and communicate these needs to the SE organization for conversion to product requirements and other system requirements. Kossiakoff and Sweet (2003, 173) discuss the importance of understanding customer needs.
- **Contracts Organizations:** These organizations interface with both customer and supplier organizations. Their role is to develop clearly stated contracts for the developer or the supplier. These contracts convey tasks and responsibilities for all SE roles of all parties. Technical specifications are attached to the contracts. Responsibilities for verification and validation are specified.
- **Supplier Management Organizations:** These organizations are responsible for selecting and managing suppliers and assuring that both contractual and technical products are in place. These organizations balance cost and risk to assure that supplier products are delivered, verified, and validated for quality product. Blanchard and Fabrycky (2005, 696-698) discuss the importance of supplier selection and agreement.

References

Works Cited

Blanchard, B. and W. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Browning, T.R. 2009. "Using the Design Structure Matrix to Design Program Organizations." In A.P. Sage and W.B. Rouse (eds.), *Handbook of Systems Engineering and Management,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Construction Management (CM) Guide. 2009. *Project Organization Types*. Accessed on September 14, 2011. Available at http://cmguide.org/archives/319.

DAU. 2001. *Systems Engineering Fundamentals*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU), U.S. Department of Defense (DoD). Accessed on September 14, 2011. Available at http://www.dau.mil/pubscats/PubsCats/SEFGuide%2001-01.pdf.

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons.

Elliott et al. 2008. *INCOSE UK Chapter Working Group on Applying Systems Engineering to In-Service Systems*. Final Report. Somerset, UK: INCOSE UK Chapter Working Group. Accessed September 6, 2011. Available at http://www.incoseonline.org.uk/Documents/Groups/InServiceSystems/is_tr_001_final_report_final_1_0.pdf.

Fasser, Y. and D. Brettner. 2002. *Management for Quality in High-Technology Enterprises*. Hoboken, NJ, USA: John Wiley & Sons.

ISO/IEC. 2000. *International standards for quality management.* Genève, Switzerland: International Organization for Standardization. ISO 9000:2000.

ISO/IEC. 2008. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2008.

Kasser, J., D. Hitchins, and T. Huynh. 2009. "Re-engineering Systems Engineering." Proceedings of the 3rd Annual Asia-Pacific Conference on Systems Engineering (APCOSE). 20-23 July 2009. Singapore.

Kossiakoff, A., and W.N. Sweet. 2003. *Systems Engineering: Principles and Practice*. Edited by A. Sage, Wiley Series in Systems Engineering and Management. Hoboken, NJ: John Wiley & Sons.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Rechtin, E. 1991. *Systems Architecting: Creating and Building Complex Systems.* Englewood Cliffs, NJ, USA: CRC Press.

Ring, J. and A.W. Wymore (eds.). 2004. *Concept of Operations (conops) of A Systems Engineering Education Community (SEEC).* Seattle, WA, USA: INCOSE Education Measurement Working Group (EMWG), INCOSE-TP-2003-015-01.

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Sheard, S. 1996. "12 Systems Engineering Roles." Paper presented at the Sixth Annual International Council on Systems Engineering (INCOSE) International Symposium. 7-11 July 1996. Boston, MA, USA.

Sheard, S. 2000. "The 12 Systems Engineering Roles Revisited." Paper presented at the INCOSE Mid-Atlantic Regional Conference. April 2000. Reston, VA, USA. p 5.2-1 - 5.2-9.

Sillitto, H. 2011a. "Unravelling Systems Engineers from Systems Engineering - Frameworks for Understanding the Extent, Variety and Ambiguity of Systems Engineering and Systems Engineers." Paper presented at the 21st Annual International Council on Systems Engineering (INCOSE) International Symposium. 20-23 June 2011. Denver, CO,

USA.

Sillitto, H. 2011b. *Sharing Systems Engineering Knowledge through INCOSE: INCOSE as An Ultra-Large-Scale System?* INCOSE *Insight.* 14(1) (April): 20.

Warfield, J. 2006. *An Introduction to Systems Science*. Washington, DC, USA: The National Academies Press, World Scientific.

Primary References

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*, 3rd ed. Hoboken, NJ, USA: John Wiley and Sons.

Kotter, J. 1995. "Leading Change: Why Transformation Efforts Fail.*"* Harvard Business Review. *73(2): 59–67.*

Sheard, S. 2000. "Systems Engineering Roles Revisited." Paper presented at the INCOSE Mid-Atlantic Regional Conference. April 5-8 2000. Reston, VA, USA. p 5.2-1 - 5.2-9.

Additional References

Blanchard, B. and W. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th edition. Upper Saddle River, NJ, USA: Prentice Hall.

Construction Management (CM) Guide. 2009. *Project Organization Types.* Accessed on September 6, 2011. Available at http://cmguide.org/archives/319.

Defense Acquisition University (DAU). 2001. *Systems Engineering Fundamentals.* Fort Belvoir, VA, USA: Defense Acquisition University Press. Accessed on September 6, 2011. Available at http:/ / www. dau. mil/ pubscats/ PubsCats/SEFGuide%2001-01.pdf.

Handy, C.B. 1985. *Understanding Organizations.* London, UK: Penguin Business.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Assessing Systems Engineering Performance of Business and Enterprises

At the project level, systems engineering (SE) measurement focuses on indicators of project and system success that are relevant to the project and its stakeholders. At the enterprise level there are additional concerns. SE governance should ensure that the performance of systems engineering within the enterprise adds value to the organization, is aligned to the organization's purpose, and implements the relevant parts of the organization's strategy.

For enterprises that are traditional businesses this is easier, because such organizations typically have more control levers than more loosely structured enterprises. The governance levers that can be used to improve performance include people (selection, training, culture, incentives), process, tools and infrastructure, and organization; therefore, the assessment of systems engineering performance in an enterprise should cover these dimensions.

Being able to aggregate high quality data about the performance of teams with respect to SE activities is certainly of benefit when trying to guide team activities. Having access to comparable data, however, is often difficult, especially in organizations that are relatively autonomous, use different technologies and tools, build products in different domains, have different types of customers, etc. Even if there is limited ability to reliably collect and aggregate data across teams, having a policy that consciously decides how the enterprise will address data collection and analysis is valuable.

Performance Assessment Measures

Typical measures for assessing SE performance of an enterprise include the following:

- Effectiveness of SE process
- Ability to mobilize the right resources at the right time for a new project or new project phase
- Quality of SE process outputs
- Timeliness of SE process outputs
- SE added value to project
- System added value to end users
- SE added value to organization
- Organization's SE capability development
- Individuals' SE competence development
- Resource utilization, current and forecast
- Productivity of systems engineers
- Deployment and consistent usage of tools and methods

How Measures Fit in the Governance Process and Improvement Cycle

Since collecting data and analyzing it takes effort that is often significant, measurement is best done when its purpose is clear and is part of an overall strategy. The "goal, question, metric" paradigm (Basili 1992) should be applied, in which measurement data is collected to answer specific questions, the answer to which helps achieve a goal, such as decreasing the cost of creating a system architecture or increasing the value of a system to a particular stakeholder. Figure 1 shows one way in which appropriate measures inform enterprise level governance and drive an improvement cycle such as the Six Sigma DMAIC (Define, Measure, Analyze, Improve, Control) model.

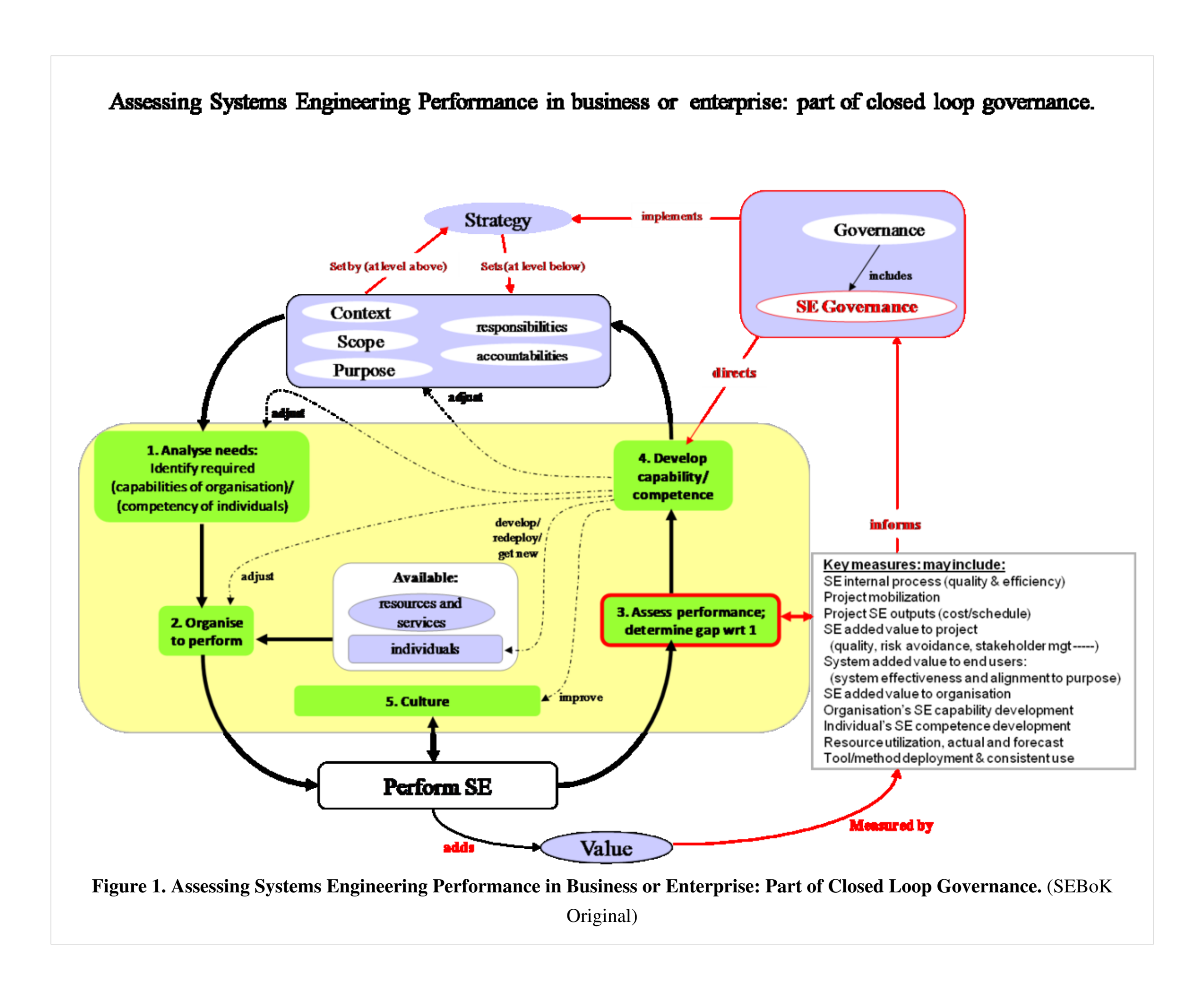


Figure 1. Assessing Systems Engineering Performance in Business or Enterprise: Part of Closed Loop Governance. (SEBoK Original)

Discussion of Performance Assessment Measures

Assessing SE Internal Process (Quality and Efficiency)

A Process (glossary) is a *"set of interrelated or interacting activities which transforms inputs into outputs."* The SEI CMMI Capability Maturity Model (SEI 2010) provides a structured way for businesses and enterprises to assess their SE processes. In the CMMI, a process area is a cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area. There are CMMI models for acquisition, for development, and for services (SEI 2010, 11). CMMI defines how to assess individual process areas against Capability Levels on a scale from 0 to 3, and overall organizational maturity on a scale from 1 to 5.

Assessing Ability to Mobilize for a New Project or New Project Phase

Successful and timely project initiation and execution depends on having the right people available at the right time. If key resources are deployed elsewhere, they cannot be applied to new projects at the early stages when these resources make the most difference. Queuing theory shows that if a resource pool is running at or close to capacity, delays and queues are inevitable.

The ability to manage teams through their lifecycle is an organizational capability that has substantial leverage on project and organizational efficiency and effectiveness. This includes being able to

- mobilize teams rapidly;
- establish and tailor an appropriate set of processes, metrics and systems engineering plans;
- support them to maintain a high level of performance;
- capitalize acquired knowledge; and
- redeploy team members expeditiously as the team winds down.

Specialists and experts are used to a review process, critiquing solutions, creating novel solutions, and solving critical problems. Specialists and experts are usually a scarce resource. Few businesses have the luxury of having enough experts with all the necessary skills and behaviors on tap to allocate to all teams just when needed. If the skills are core to the business' competitive position or governance approach, then it makes sense to manage them through a governance process that ensures their skills are applied to greatest effect across the business.

Businesses typically find themselves balancing between having enough headroom to keep projects on schedule when things do not go as planned and utilizing resources efficiently.

Project SE Outputs (Cost, Schedule, Quality)

Many SE outputs in a project are produced early in the life cycle to enable downstream activities. Hidden defects in the early phase SE work products may not become fully apparent until the project hits problems in integration, verification and validation, or transition to operations. Intensive peer review and rigorous modeling are the normal ways of detecting and correcting defects in and lack of coherence between SE work products.

Leading indicators could be monitored at the organizational level to help direct support to projects or teams heading for trouble. For example, the INCOSE Leading Indicators report (Roedler et al. 2010) offers a set of indicators that is useful at the project level. Lean Sigma provides a tool for assessing benefit delivery throughout an enterprise value stream. Lean Enablers for Systems Engineering are now being developed (Oppenheim et al. 2010). An emerging good practice is to use Lean Value Stream Mapping (glossary) to aid the optimization of project plans and process application.

In a mature organization, one good measure of SE quality is the number of defects that have to be corrected "out of phase"; i.e., at a later phase in the life cycle when the defect was introduced. This gives a good measure of process performance and the quality of SE outputs. Within a single project, the Work Product Approval, Review Action Closure, and Defect Error trends contain information that allows residual defect densities to be estimated (Roedler et al. 2010; Davies and Hunter 2001)

Because of the leverage of front-end SE on overall project performance, it is important to focus on quality and timeliness of SE deliverables (Woodcock 2009).

SE Added Value to Project

SE that is properly managed and performed should add value to the project in terms of quality, risk avoidance, improved coherence, better management of issues and dependencies, right-first-time integration and formal verification, stakeholder management, and effective scope management. Because quality and quantity of SE are not the only factors that influence these outcomes, and because the effect is a delayed one (good SE early in the project pays off in later phases) there has been a significant amount of research to establish evidence to underpin the asserted

benefits of SE in projects.

A summary of the main results is provided in the Economic Value of Systems Engineering article.

System Added Value to End Users

System added value to end users depends on system effectiveness and on alignment of the requirements and design to the end users' purpose and mission. System end users are often only involved indirectly in the procurement process.

Research on the value proposition of SE shows that good project outcomes do not necessarily correlate with good end user experience. Sometimes systems developers are discouraged from talking to end users because the acquirer is afraid of requirements creep. There is experience to the contrary, that end user involvement can result in more successful and simpler system solutions.

Two possible measures indicative of end user satisfaction are

1. The use of user-validated mission scenarios (both nominal and "rainy day" situations) to validate requirements, drive trade-offs and organize testing and acceptance;
2. The use of Technical Performance Measure (TPM) (glossary) to track critical performance and non-functional system attributes directly relevant to operational utility. The INCOSE SE Leading Indicators Guide (Roedler et al. 2010, 10 and 68) defines "technical measurement trends" as *"Progress towards meeting the Measure of Effectiveness (MoE) (glossary) / Measure of Performance (MoP) (glossary) / Key Performance Parameters (KPPs) and Technical Performance Measure (TPM) (glossary)"*. A typical TPM progress plot is shown in Figure 2.

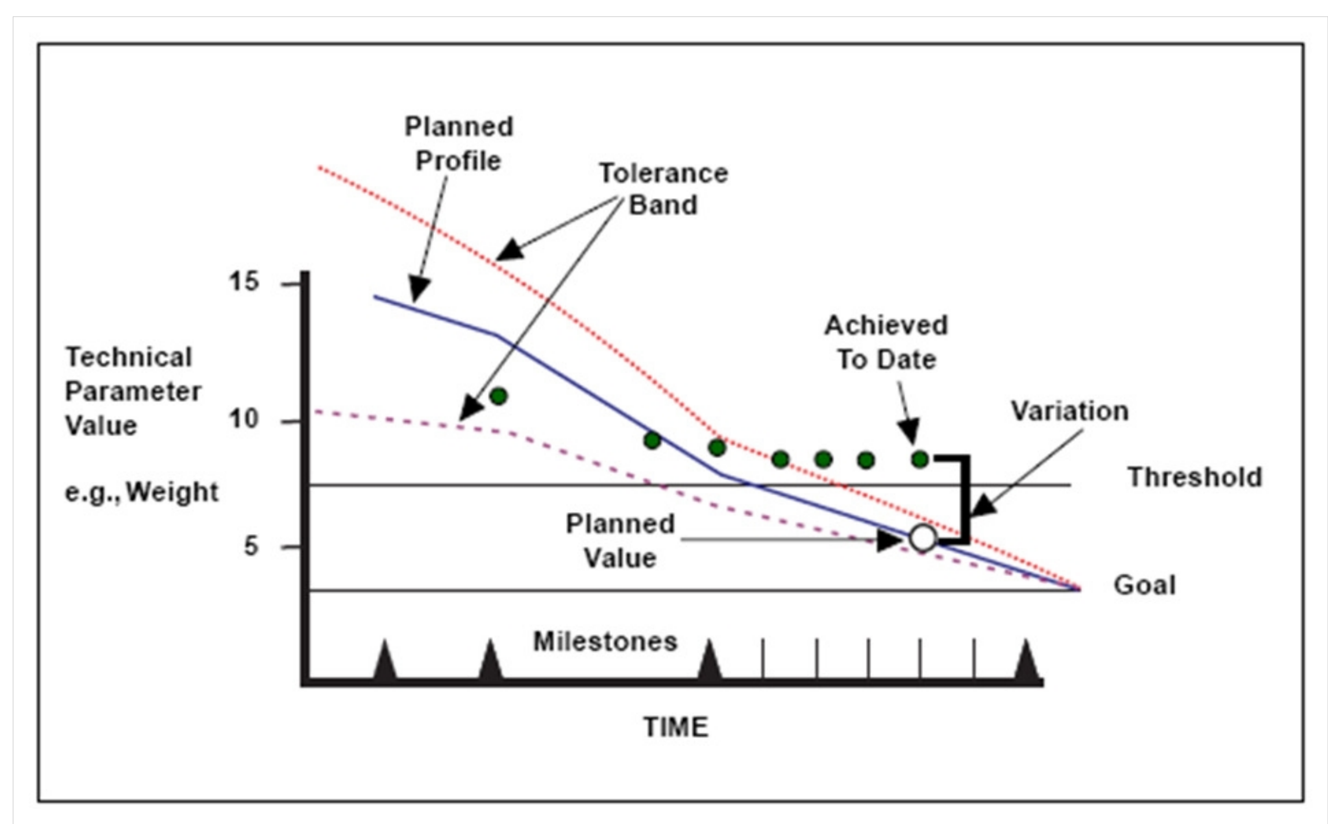


Figure 2. Technical Performance Measure (TPM) Tracking (Roedler et al. 2010). This material is reprinted with permission from the International Council on Systems Engineering (INCOSE). All other rights are reserved by the copyright owner.

SE Added Value to Organization

SE at the business/enterprise level aims to develop, deploy and enable effective SE to add value to the organization’s business. The SE function in the business/enterprise should understand the part it has to play in the bigger picture and identify appropriate performance measures - derived from the business or enterprise goals, and coherent with those of other parts of the organization - so that it can optimize its contribution.

Organization's SE Capability Development

The CMMI (SEI 2010) provides a means of assessing the process capability and maturity of businesses and enterprises. The higher CMMI levels are concerned with systemic integration of capabilities across the business or enterprise.

CMMI measures one important dimension of capability development, but CMMI maturity level is not a direct measure of business effectiveness unless the SE measures are properly integrated with business performance measures. These may include bid success rate, market share, position in value chain, development cycle time and cost, level of innovation and re-use, and the effectiveness with which SE capabilities are applied to the specific problem and solution space of interest to the business.

Individuals' SE Competence Development

Assessment of Individuals' SE competence development is described in Assessing Individuals.

Resource Utilization, Current and Forecast

Roedler et al. (2010, 58) offer various metrics for staff ramp-up and use on a project. Across the business or enterprise, key indicators include the overall manpower trend across the projects, the stability of the forward load, levels of overtime, the resource headroom (if any), staff turnover, level of training, and the period of time for which key resources are committed.

Deployment and Consistent Usage of Tools and Methods

It is common practice to use a range of software tools in an effort to manage the complexity of system development and in-service management. These range from simple office suites to complex logical, virtual reality and physics-based modeling environments.

Deployment of SE tools requires careful consideration of purpose, business objectives, business effectiveness, training, aptitude, method, style, business effectiveness, infrastructure, support, integration of the tool with the existing or revised SE process, and approaches to ensure consistency, longevity and appropriate configuration management of information. Systems may be in service for upwards of 50 years, but storage media and file formats that are 10-15 years old are unreadable on most modern computers. It is desirable for many users to be able to work with a single common model; it can be that two engineers sitting next to each other using the same tool use sufficiently different modeling styles that they cannot work on or re-use each others' models.

License usage over time and across sites and projects is a key indicator of extent and efficiency of tool deployment. More difficult to assess is the consistency of usage. Roedler et al. (2010, 73) recommend metrics on "facilities and equipment availability".

Practical Considerations

Assessment of SE performance at the business/enterprise level is complex and needs to consider soft issues as well as hard issues. Stakeholder concerns and satisfaction criteria may not be obvious or explicit. Clear and explicit reciprocal expectations and alignment of purpose, values, goals and incentives help to achieve synergy across the organization and avoid misunderstanding.

"What gets measured gets done." Because metrics drive behavior, it is important to ensure that metrics used to manage the organization reflect its purpose and values, and that they do not drive perverse behaviors (Roedler et al. 2010).

Process and measurement cost money and time, so it is important to get the right amount of process definition and the right balance of investment between process, measurement, people and skills. Any process flexible enough to allow innovation will also be flexible enough to allow mistakes. If process is seen as excessively restrictive or prescriptive, in an effort to prevent mistakes it may inhibit innovation and demotivate the innovators, leading to excessive risk avoidance.

It is possible for a process improvement effort to become an end in itself rather than a means to improve business performance (Sheard 2003). To guard against this, it is advisable to remain clearly focused on purpose (Blockley and Godfrey 2000) and on added value (Oppenheim et al. 2010) as well as to ensure clear and sustained top management commitment to driving the process improvement approach to achieve the required business benefits. Good process improvement is as much about establishing a performance culture as about process.

> *The Systems Engineering process is an essential complement to, and is not a substitute for, individual skill, creativity, intuition, judgment etc. Innovative people need to understand the process and how to make it work for them, and neither ignore it nor be slaves to it. Systems Engineering measurement shows where invention and creativity need to be applied. SE process creates a framework to leverage creativity and innovation to deliver results that surpass the capability of the creative individuals – results that are the emergent properties of process, organisation, and leadership.* (Sillitto 2011)

References

Works Cited

Basili, V. 1992. "Software Modeling and Measurement: The Goal/Question/Metric Paradigm" Technical Report CS-TR-2956. University of Maryland: College Park, MD, USA. Accessed on August 28, 2012. Available at http://www.cs.umd.edu/~basili/publications/technical/T78.pdf.

Blockley, D. and P. Godfrey. 2000. *Doing It Differently – Systems For Rethinking Construction*. London, UK: Thomas Telford Ltd.

Davies, P. and N. Hunter. 2001. "System Test Metrics on a Development-Intensive Project." Paper presented at the 11th Annual International Council on System Engineering (INCOSE) International Symposium. 1-5 July 2001. Melbourne, Australia.

Oppenheim, B., E. Murman, and D. Sekor. 2010. *Lean Enablers for Systems Engineering*. Systems Engineering. 14(1). New York, NY, USA: Wiley and Sons, Inc.

Roedler, G. D. Rhodes, H. Schimmoller, and C. Jones (eds.). 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. January 29, 2010, Published jointly by LAI, SEARI, INCOSE, and PSM. INCOSE-TP-2005-001-03. Accessed on September 14, 2011. Available at http://seari.mit.edu/documents/SELI-Guide-Rev2.pdf.

SEI. 2010. *CMMI for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute/Carnegie Mellon University. CMU/SEI-2010-TR-033. Accessed on September 14, 2011. Available at http://www.sei.cmu.edu/reports/10tr033.pdf.

Sheard, S, 2003. "The Lifecycle of a Silver Bullet." *Crosstalk: The Journal of Defense Software Engineering.* (July 2003). Accessed on September 14, 2011. Available at http://www.crosstalkonline.org/storage/issue-archives/2003/200307/200307-Sheard.pdf.

Sillitto, H. 2011. Panel on "People or Process, Which is More Important". Presented at the 21st Annual International Council on Systems Engineering (INCOSE) International Symposium. 20-23 June 2011. Denver, CO, USA.

Woodcock, H. 2009. "Why Invest in Systems Engineering." INCOSE UK Chapter. Z-3 Guide, Issue 3.0. March 2009. Accessed on September 14, 2011. Available at http://www.incoseonline.org.uk/Documents/zGuides/Z3_Why_invest_in_SE.pdf.

Primary References

Basili, V. 1992. "Software Modeling and Measurement: The Goal/Question/Metric Paradigm". College Park, MD, USA: University of Maryland. Technical Report CS-TR-2956. Accessed on August 28, 2012. Available at http://www.cs.umd.edu/~basili/publications/technical/T78.pdf.

Frenz, P., et al. 2010. *Systems Engineering Measurement Primer: A Basic Introduction to Measurement Concepts and Use for Systems Engineering,* version 2.0. San Diego, CA, USA: International Council on System Engineering (INCOSE). INCOSE-TP-2010-005-02.

Oppenheim, B., E. Murman, and D. Sekor. 2010. *Lean Enablers for Systems Engineering*. Systems Engineering. 14(1). New York, NY, USA: Wiley and Sons, Inc.

Roedler, G., D. Rhodes, H. Schimmoller, and C. Jones (eds.). 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. January 29, 2010, Published jointly by LAI, SEARI, INCOSE, PSM. INCOSE-TP-2005-001-03. Accessed on September 14, 2011. Available at http://seari.mit.edu/documents/SELI-Guide-Rev2.pdf.

Additional References

Jelinski, Z. and P.B. Moranda. 1972. "Software Reliability Research". In W. Freiberger. (ed.), *Statistical Computer Performance Evaluation.* New York, NY, USA: Academic Press. p. 465-484.

Alhazmi O.H. and Y.K. Malaiya. 2005. *Modeling the Vulnerability Discovery Process*. 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05). 8-11 November 2005. Chicago, IL, USA.

Alhazmi, O.H. and Y.K. Malaiya. 2006. "Prediction Capabilities of Vulnerability Discovery Models." Paper presented at Annual Reliability and Maintainability Symposium (RAMS). 23-26 January 2006. p 86-91. Newport Beach, CA, USA. Accessed on September 14, 2011. Available at http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1677355&isnumber=34933.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Developing Systems Engineering Capabilities within Businesses and Enterprises

The pursuit of continuous improvement is a constant for many organizations. The description of Toyota (Morgan and Liker 2006), the Lean principle of “pursue perfection” (Oppenheim et al. 2010), and the principle of “don’t let up” (Kotter 1995), all drive a need for continuous improvement.

The ability to manage teams through their lifecycle - mobilize teams rapidly, establish and tailor an appropriate set of processes, metrics and systems engineering plans, support them to maintain a high level of performance, capitalize acquired knowledge and redeploy team members expeditiously as the team winds down - is a key organizational competence that has substantial leverage on project and organizational efficiency and effectiveness.

The enterprise provides teams with the necessary resources, background information, facilities, cash, support services, tooling, etc. It also provides a physical, cultural and governance environment in which the teams can be effective. The key functions of the enterprise include generating and maintaining relevant resources, allocating them to teams, providing support and governance functions, maintaining expertise and knowledge (on process, application domain and solution technologies), securing the work that teams perform, organizing finance, and maintaining the viability of the enterprise.

For improvements to persist, they must reside in the enterprise rather than just the individuals, so the improvements can endure as personnel leave. This is reflected in the Capability Maturity Model Integrated (CMMI) (SEI 2010) progression from a "hero culture" to a "quantitatively managed and optimizing process".

This topic outlines the issues to be considered in capability development and organizational learning.

Overview

Figure 1 shows an "analyze - organize - perform - assess - develop" cycle, which is essentially a reformulation of the Deming (1994) PDCA (Plan Do Check Act) cycle. The analysis step should cover both current and future needs, as far as these can be determined or predicted. Goals and performance assessment, as discussed in Assessing Systems Engineering Performance of Business and Enterprises, can be based on a number of evaluation frameworks, such as direct measures of business performance and effectiveness and the CMMI capability maturity models. There is evidence that many organizations find a positive correlation between business performance and CMMI levels (SEI 2010). This is discussed further in the Economic Value of Systems Engineering.

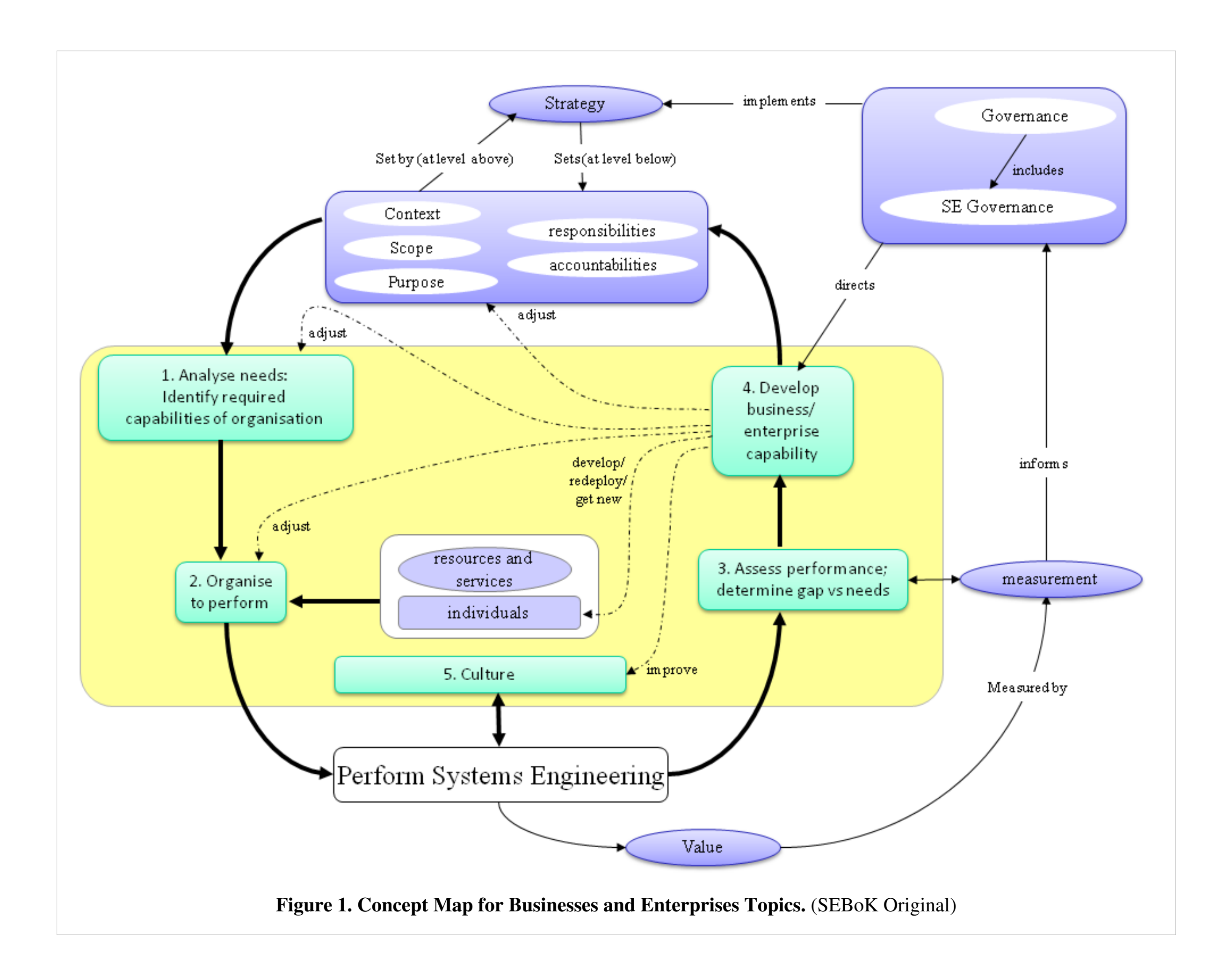


Figure 1. Concept Map for Businesses and Enterprises Topics. (SEBoK Original)

Change Levers

SE managers have a number of possible change levers they can use to develop SE capabilities. The amount of time delay between moving a lever and seeing the effect varies with the type of level, size of the enterprise, culture of the enterprise, and other factors.

Adjust Context, Scope, Purpose, Responsibility, Accountability Business Enterprise

If the other change levers cannot achieve the desired effect, the business or enterprise may have to renegotiate its contribution to the higher level strategy and mission.

Review and Adjust Required Capabilities

In the initial analysis the needed capability may have been over- or under-estimated. The need should be re-evaluated after each rotation of the cycle to make sure the planning assumptions are still valid.

Adjust Organization within Business Enterprise

Adjusting organization and responsibilities so that *"the right people are doing the right things"*, and ensuring that the organization is making full use of their knowledge and skills, is often the easiest change to make (and the one that may have the quickest effect).

A potential risk is that too much organizational churn disrupts relationships and can destabilize the organization and damage performance. Process improvement can be set back by an ill-considered re-organization and can jeopardize any certifications the organization has earned which demonstrate its process capability or performance.

Develop/Train/Redeploy/Get New Resources, Services and Individuals

Resources, services and individuals may include any of the components of organizational SE capability listed in Organizing Business and Enterprises to Perform Systems Engineering.

Levers include subcontracting elements of the work, improving information flows, upgrading facilities, and launching short-term training and/or long term staff development programs. Many organizations consider how they approach these improvements to be proprietary, but organizations such as NASA offer insight on their APPEL website (NASA 2012).

Development of individuals is discussed in Enabling Individuals.

Improve Culture

Culture change is very important, very powerful, but needs to be handled as a long-term game and given long term commitment.

Adjust and Improve Alignment of Measures and Metrics

Measurement drives behavior. Improving alignment of goals and incentives of different parts of the business/enterprise so that everyone works to a common purpose can be a very effective and powerful way of improving business/enterprise performance. This alignment does require some top-down guidance, perhaps a top-down holistic approach, considering the business/enterprise as a system with a clear understanding of how the elements of enterprise capability interact to produce synergistic value (See Assessing Systems Engineering Performance of Business and Enterprises). It is commonly reported that as an organization improves its processes with respect to the CMMI, its approach to metrics and measurement has to evolve.

Change Methods

Doing Everyday Things Better

There is a wealth of sources and techniques, including Kaizen, Deming PDCA (Deming 1994), Lean (Womack and Jones 2003, Oppenheim et al. 2010), Six-Sigma (Harry 1997), and CMMI.

Value stream mapping is a powerful Lean technique to find ways to improve flow and handovers at interfaces.

Managing Technology Readiness

In high-technology industries many problems are caused by attempting to transition new technologies into products and systems before the technology is mature; to make insufficient allowance for the effort required to make the step from technology demonstration to reproducible and dependable performance in a product; or to overestimate the re-usability of an existing product. NASA's TRL (Technology Readiness Level) construct, first proposed by John Mankins in 1995 (Mankins 1995), is widely and successfully used to understand and mitigate technology transition risk. Several organizations beyond NASA, such as the U.S. Department of Defense, even have automation to aid engineers in evaluating technology readiness.

Variations on TRL have even emerged, such as System Readiness Levels (SRL) (Sauser et al. 2006), which recognize that the ability to successfully deliver systems depends on much more than the maturity of the technology base used to create those systems; e.g., there could be surprising risks associated with using two technologies that are relatively mature in isolation, but have never been integrated together before.

Planned Change: Standing Up or Formalizing SE in an Organization

Planned change may include:

- introducing SE to a business (Farncombe and Woodcock 2009);
- improvement/transformation;
- formalizing the way a business or project does SE;
- dealing with a merger/demerger/major re-organization;
- developing a new generation or disruptive product, system, service or product line (Christensen 1997);
- entering a new market; and
- managing project lifecycle transitions: start-up, changing to the next phase of development, transition to manufacture/operation/support, wind down and decommissioning.

CMMI is widely used to provide a framework for planned change in a systems engineering context. Planned change needs to take a holistic approach considering people (knowledge, skills, culture, ability and motivation), process, measurement and tools as a coherent whole. It is now widely believed that tools and process are not a substitute for skills and experience. Instead, they merely provide a framework in which skilled and motivated people can be more effective. So change should start with people rather than with tools.

Before a change is started, it is advisable to baseline the current business performance and SE capability and establish metrics that will show early on whether the change is achieving the desired effect.

Responding to Unforeseen Disruption

Unforeseen disruptions may be internally or externally imposed. Externally imposed disruptions may be caused by

- the customer - win/lose contract, mandated teaming or redirection;
- competitors - current offering becomes less/more competitive, a disruptive innovation may be launched in market; or
- governance and regulatory changes - new processes, certification, safety or environmental standards.

Internal or self-induced disruptions may include

- a capability drop-out due to loss of people, facilities, financing;
- product or service failure in operation or disposal; or
- strategy change (e.g. new CEO, response to market dynamics, or a priority override).

Embedding Change

In an SE context, sustained effort is required to maintain improvements such as higher CMMI levels, Lean and Safety cultures, etc., once they are achieved. There are several useful change models, including Kotter's 8 phases of change (Kotter 1995):

1. Establish a sense of urgency;
2. Create a coalition;
3. Develop a clear vision;
4. Share the vision;
5. Empower people to clear obstacles;
6. Secure short term wins;
7. Consolidate and keep moving; and
8. Anchor the change.

The first six steps are the easy ones. The Chaos Model (Zuijderhoudt 1990; 2002) draws on complexity theory to show that regression is likely if the short term wins are not consolidated, institutionalized and anchored. This explains the oft-seen phenomenon of organizations indulging in numerous change initiatives, none of which sticks because attention moves on to the next before the previous one is anchored.

Change Management Literature

SE leaders (directors, functional managers, team leaders and specialists) have responsibilities, and control levers to implement them, that vary depending on their organization's business model and structure. A great deal of their time and energy is spent managing change in pursuit of short, medium and long term organizational goals: "doing everyday things better"; making change happen, embedding change and delivering the benefit; and coping with the effects of disruptions. Mergers, acquisitions and project start-ups, phase changes, transitions from "discovery" to "delivery" phase, transition to operation, sudden change in level of funding, can all impose abrupt changes on organizations that can destabilize teams, processes, culture and performance. Table 1 below provides both the general management literature and specific systems engineering knowledge.

Table 1. Change Management – Business and SE References. (SEBoK Original)

Area	Business references	SE references
Doing every day things better	Kaizen; Lean (Womack and Jones 2003); 6-Sigma (Harry 1997) Four competencies of Learning Organisation – absorb, diffuse, generate, exploit (Sprenger and Ten Have 1996) Covey's seven habits of very effective people (Covey 1989)	CMMI Forsberg & Mooz, Visualizing project management (Forsberg and Mooz 2005) INCOSE IEWG "Conops for a Systems Engineeriing Educational Community" (Ring and Wymore 2004) INCOSE Lean Enablers for SE (Oppenhein et al. 2010)
Dealing with unplanned disruption	Mitroff, managing crises before they happen (Mitroff and Anagnos 2005); Shell, Scenario Planning (Wack 1985; Ringland 1988)	Scott Jackson, architecting resilient systems (Jackson 2010) Design principles for ultra-large-scale systems (Sillitto 2010)
Driving disruptive innovation	Christensen's Innovator's Dilemma (Christensen 1997) Mintzberg "Rise and fall of strategic planning", (Mintzberg 2000) BS7000, Standard for innovation management (BSI 2008)	
Exploiting unexpected opportunities	Mintzberg, rise and fall of strategic planning (Mintzberg 2000) Mission Command (military), Auftragstechnik (Bungay 2002, 32)	Architecting for Flexibility and Resilience (Jackson 2010) Open system architectures; Lean SE; (Oppenheim et al. 2010) Agile methodologies
Implementing and embedding planned change	Kotter's eight phases of change (Kotter 1995), Berenschot's seven forces (ten Have et al. 2003) Levers of control (Simons 1995) – tension between control, creativity, initiative and risk taking Chaos model, "complexity theory applied to change processes in organisations"; (Zuiderhoudt and Ten Have 1999) Business Process Re-engineering (Hammer and Champy 1993) Senge's 5th discipline (Senge 2006) Change Quadrants (Amsterdam 1999)	"Doing it differently - systems for rethinking construction" (Blockley and Godfrey 2000) INCOSE UK Chapter Z-guides: • Z-2, introducing SE to an organisation (Farncombe and Woodcock 2009); • Z-7, Systems Thinking (Godfrey and Woodcock 2010)
Understanding peoples' motivation, behaviour	Maslow's hierarchy of needs Myers-Briggs Type Indicator; NLP (Neuro-Linguistic Programming) (See for example: Knight 2009) Socio-technical organisation (Taylor and Felten 1993) Core quadrants, (Offman 2001)	INCOSE Intelligent Enterprise Working Group – "enthusiasm", stretch goals (Ring and Wymore 2004) Sommerville, Socio Technical Systems Engineering, Responsibility Mapping (Sommerville et al. 2009)
Understanding culture	Cultural Dimensions, (Hofstede 1994) Compliance Typology (Etzione 1961)	

Helping individuals cope with change	5 C's of individual change, and Rational/emotional axes, Kets De Vries, quoted in "key management models" (Ten Have et al. 2003)	Relationships made easy (Fraser 2010) – rational/emotional, NLP and other methods

References

Works Cited

Blockley, D. and P. Godfrey. 2000. *Doing It Differently – Systems For Rethinking Construction.* London, UK: Thomas Telford, Ltd.

Bungay, S. 2002. *Alamein*. London, UK: Aurum press. First published 2002, Paperback 2003.

BSI. 2008. *Design Management Systems. Guide to Managing Innovation.* London, UK: British Standards Institution (BSI). BS 7000-1:2008.

Christensen, C. 1997. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail.* Cambridge, MA, USA: Harvard Business School Press.

Covey, S.R. 1989. *The Seven Habits of Highly Effective People.* Also released as a 15th Anniversary Edition (2004). New York, NY, USA: Simon & Schuster, 1989.

Deming, W.E. 1994. *The New Economics.* Cambridge, MA, USA: Massachusetts Institute of Technology, Centre for Advanced Educational Services.

Etzione, A. 1961. *A Comparative Analysis of Complex Organizations. On Power, Involvement and their Correlates.* New York, NY, USA: The Free Press of Glencoe, Inc.

Farncombe, A. and H. Woodcock. 2009. "Enabling Systems Engineering." Somerset, UK: INCOSE UK Chapter. *Z-2 Guide*, Issue 2.0 (March 2009). Accessed September 14, 2011. Available at http://www.incoseonline.org.uk/Documents/zGuides/Z2_Enabling_SE.pdf.

Forsberg, K. and H. Mooz. 2005. *Visualizing Program Management, Models and Frameworks for Mastering Complex Systems,* 3rd ed. New York, NY, USA: Wiley and Sons, Inc.

Fraser, D. 2010. *Relationships Made Easy: How To Get on with The People You Need to Get on With...and Stay Friends with Everyone Else.* Worcestershire, UK: Hothive Publishing.

Godfrey, P. and H. Woodcock. 2010. "What is Systems Thinking?" Somerset, UK: INCOSE UK Chapter, *Z-7 Guide*, Issue 1.0 (March 2010). Accessed on September 7, 2011. Available at http://www.incoseonline.org.uk/Documents/zGuides/Z7_Systems_Thinking_WEB.pdf.

Hammer, M. and J.A. Champy. 1993. *Reengineering the Corporation: A Manifesto for Business Revolution.* New York, NY, USA Harper Business Books.

Harry, M.J. 1997. *The Nature of Six Sigma Quality*. Schaumburg, IL, USA: Motorola University Press.

Hofstede, G. 1984. *Culture's Consequences: International Differences in Work-Related Values.* Newbury Park, CA, USA and London, UK: Sage Publications Inc.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions.* A. P. Sage (ed.). Wiley Series in Systems Engineering and Management. New York, NY, USA: Wiley & Sons, Inc.

Knight, S. 2009. *NLP at Work - Neuro Linguistic Programming - The Essence of Excellence,* 1st edition 1995. 3rd edition 2009. London, UK and Boston, MA, USA: Nicholas Brealey Publishing.

Kotter, J. 1995. "Leading Change: Why Transformation Efforts Fail". *Harvard Business Review.* (March-April 1995).

Mintzberg, H. 2000. *The Rise and Fall of Strategic Planning*. Upper Saddle River, NJ, USA: Pearson Education.

Mitroff, I. and G. Anagnos. 2005.*Managing Crises Before They Happen: What Every Executive and Manager Needs to Know about Crisis Management*. New York, NY, USA: AMACOM Press.

Morgan, J. and J. Liker. 2006. *The Toyota Product Development System: Integrating People, Process and Technology.* New York, NY, USA: Productivity Press.

NASA. 2012. "APPEL: Academy of Program/Project & Engineering Leadership." Accessed on September 9, 2012. Available at http://www.nasa.gov/offices/oce/appel/seldp/nasa-se/index.html.

Offman, D.D. 2001. *Inspiration and Quality in Organizations,* (Original title (Dutch): *Bezieling en kwaliteit in organisaties*), 12th Edition. Utrecht, The Netherlands: Kosmos-Z&K.

Oppenheim, B., E. Murman, and D. Sekor. 2010. *Lean Enablers for Systems Engineering*. Systems Engineering. 14(1). New York, NY, USA: Wiley and Sons, Inc.

Ring, J. and A.W. Wymore (eds.) 2004. *Concept of Operations (conops) of A Systems Engineering Education Community (SEEC).* Seattle, WA, USA: INCOSE Education Measurement Working Group (EMWG), INCOSE-TP-2003-015-01.

Ringland, G. 1998. *Scenario Planning: Managing for the Future.* New York, NY, USA: Wiley and Sons, Inc.

Sauser, B., Verma, D., Ramirez-Marquez, J. and Gove, R. 2006. From TRL to SRL: The Concept of System Readiness Levels. Proceedings of the Conference on Systems Engineering Research (CSER), Los Angeles, CA.

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Senge, P.M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization,* 2nd ed. New York, NY, USA: Doubleday/Currency.

Simons, R. 1995. *Levers of Control, How Managers Use Innovative Control Systems to Drive Strategic Renewal.* Boston, MA, USA: Harvard Business School Press.

Sillitto, H. 2010. "Design Principles for Ultra-Large Scale Systems." Paper in 20th Annual International Council on Systems Engineering (INCOSE) International Symposium. 12-15 July 2010. Chicago, IL, USA. (Reprinted in The Singapore Engineer, IES, April 2011).

Sommerville, I., R. Lock, T. Storer, and J.E. Dobson. 2009. "Deriving Information Requirements from Responsibility Models. *Paper in the 21st International Conference on Advanced Information Systems Engineering (CAiSE). June 2009. Amsterdam, Netherlands. p. 515-529.*

Sprenger, C. and S. Ten Have. 1996. "4 Competencies of a Learning Organisation." (Original title (Dutch): *Kennismanagement als moter van delerende organisatie*), *Holland Management Review,* (Sept–Oct): 73–89.

Taylor, J.C.and D.F. Felten. 1993. *Performance by Design: Sociotechnical Systems in North America.* Englewood Cliffs, NJ, USA: Formerly Prentice Hall, Pearson Education Ltd.

ten Have, S., W.T. Have, F. Stevens, and M. van der Elst. 2003. *Key Management Models - The Management Tools and Practices That Will Improve Your Business.* Upper Saddle River, NJ, USA: Pearson Education Ltd. (Formerly Prentice Hall).

Wack, P. 1985. "Scenarios: Uncharted Waters Ahead." *Harvard Business Review*. (September-October 1985).

Womack, J. and D. Jones. 2003. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation,* Revised Edition. New York, NY, USA: Simon & Schuster.

Zuiderhoudt, W. and B. Ten Have. 1999. *Complexity Theory Applied to Change Processes in Organisations*.

Primary References

Kotter, J. 1995. *Leading Change: Why Transformation Efforts Fail. Harvard Business Review.* (March-April 1995).

Oppenheim, B., E. Murman, and D. Sekor. 2010. *Lean Enablers for Systems Engineering. Systems Engineering.* 14(1). New York, NY, USA: Wiley and Sons, Inc.

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development*, version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Senge, P.M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*, 2nd ed. New York, NY, USA: Doubleday/Currency.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Culture

Establishing and managing cultures, values, and behaviors is a critical aspect of systems engineering, especially in the context of deploying SE within an organization (Fasser and Brettner 2002). The Columbia Accident Investigation Report (NASA 2003, 101), defines *culture (glossary)* as *"the basic values, norms, beliefs, and practices that characterize the functioning of a particular institution."*

Stable safety and process cultures are key to effective SE, and can be damaged by an overly-rapid pace of change, a high degree of churn (see the Nimrod Crash Report, Haddon-Cave 2009), or by change that engineers perceive as arbitrarily imposed by management (see Challenger, discussed below). On the other hand, a highly competitive, adversarial or "blame" culture can impede the free flow of information and disrupt synergies in the workplace.

In the multi-national, multi-business, multi-discipline collaborative projects becoming increasingly prevalent in SE, these factors take on greater importance.

Effective handling of cultural issues is a major factor in the success or failure of SE endeavors.

Systems Thinking and the Culture of the Learning Organization

Improving SE efficiency and effectiveness can be the goal of cultural change. This kind of culture change encourages people to learn to think and act in terms of systems, organizations and their enterprises; and, to take a systems approach as described in Overview of Systems Approaches in Part 2, and by Lawson (2010). See the knowledge area Systems Thinking.

Attaining a *learning organization* culture can be another goal of cultural change. And once the learning organization exists, cultural change in general becomes easier to accomplish.

A learning organization aims to absorb, diffuse, generate, and exploit knowledge (Sprenger and Have 1996). Organizations need to manage formal information and facilitate the growth and exploitation of tacit knowledge. They should learn from experience and create a form of *corporate memory* – including process, problem domain and solution space knowledge, and information about existing products and services. Fassner and Brettner (2002, 122-124) suggest that *shared mental models* are a key aspect of corporate knowledge and culture.

A learning organization culture is enabled by disciplines such as

- **personal mastery** where a person continually clarifies and deepens personal vision, focuses energy upon it, develops patience in seeking it so as to view reality in an increasingly objective way;
- **mental models** where people appreciate that mental models do indeed occupy their minds and shape their actions;
- **shared vision** where operating values and sense of purpose are shared to establish a basic level of mutuality; and
- **team learning** where people's thoughts align, creating a feeling that the team as a whole achieves something greater than the sum of what is achieved by its individual members.

Systems thinking supports these four disciplines, and in so doing becomes the **fifth discipline** and plays a critical role in promoting the learning organization (Senge et al. 1994).

Cultural Shortfalls and How to Change them

Cultural shortfalls that are injurious to a system are described as negative paradigms (glossary) by Jackson (2010) and others. For example, a cultural reluctance to identify true risks (glossary) is the hallmark of the **Risk Denial** paradigm as seen in the Challenger and Columbia cases. When individuals believe a system is safe that in fact is not, that is the **Titanic Effect** paradigm, which is of course named for the ocean liner catastrophe of 1912.

Approaches to Change

Jackson and Erlick (Jackson 2010, 91-119) have found that there is a lack of evidence that a culture can be changed from a success point of view. However, they do suggest the Community of Practice (Jackson 2010, 110-112), an approach founded on the principles of organizational psychology, and discuss the pros and cons of other approaches to culture change, including training, coaching, Socratic teaching, use of teams, independent reviews, standard processes, rewards and incentives, use of cost and schedule margins, reliance on a charismatic executive, and management selection. Shields (2006) provides a similarly comprehensive review.

The Columbia Accident (NASA 2003) and the Triangle fire (NYFIC 1912) official reports, among many others, call for cultural issues to be addressed through improved leadership, usually augmented by the more objective approach of auditing. One form of auditing is the Independent Technical Authority, which

- is separate from the program organization;
- addresses only technical issues, not managerial ones; and
- has the right to take action to avoid failure, including by vetoing launch decisions.

An Independent Technical Authority cannot report to the program manager of the program in question, and it may be formulated within an entirely separate business or enterprise which can view that program objectively. The point of these stipulations is to insure that the Independent Technical Authority is indeed independent.

Management and leadership experts have identified ways to lead cultural change in organizations, apart from specifically safety-related cultural change. For example, Gordon (1961) in his work on the use of analogical reasoning called synectics is one of several who emphasize creative thinking. Kotter (1995) advocates a series of steps to transform an organization.

How Culture Manifests in Individuals and Groups

As a community's physical, social, and religious environment changes over the generations, cultural beliefs, values, and customs evolve in response, albeit at a slower pace.

Helmreich and Merritt describe the effects of cultural factors in the context of aviation safety, and suggest implications for safety cultures in other domains such as medicine. See (Helmreich and Merritt, 2000) and other writings by the same authors.

We can describe the cultural orientation of an individual in terms of

- national and/or ethnic culture;

- professional culture; and
- organizational culture.

Some particulars of these aspects of culture are sketched below.

National and/or Ethnic Culture

A product of factors such as heritage, history, religion, language, climate, population density, availability of resources, and politics, national culture is acquired in one's formative years and is difficult to change. National culture affects attitudes, behavior, and interactions with others.

National culture may help determine how a person handles or reacts to

- rules and regulations;
- uncertainty; and
- display of emotion, including one’s own.

National culture may also play a role in whether a person

- communicates in a direct and specific style, or the opposite;
- provides leadership in a hierarchical manner, or a consultative one; and
- accepts decisions handed down in superior–inferior relationships, or question them.

Professional Culture

Professional culture acts as an overlay to ethnic or national culture, and usually manifests in a sense of community and in bonding based on a common identity (Helmreich and Merritt 2000). Well-known examples of professional cultures include those of medical doctors, airline pilots, teachers, and the military.

Elements of professional culture may include

- a shared professional jargon
- binding norms for behavior
- common ethical values
- self-regulation
- barriers to entry like selectivity, competition and training
- institutional and/or individual resistance to change
- prestige and status, sometimes expressed in badges or uniforms
- stereotyped notions about members of the profession, in general and/or based on gender

Particularly important elements of professional culture (for example, those that affect safety or survivability) need to be inculcated by extensive training and reinforced at appropriate intervals.

Organizational Culture

An organization's culture builds up cumulatively, determined by factors like its leadership, products and services, relationships with competitors, and role in society.

Compared with one another, organizational cultures are not standardized because what works in one organization seldom works in another. Even so, strength in the following elements normally engenders a strong organizational culture:

- corporate identity;
- leadership;
- morale and trust;
- teamwork and cooperation;
- job security;

- professional development and training;
- empowerment of individuals; and
- confidence, for example in quality and safety practices, or in management communication and feedback.

When the culture of the people in an organization is considered as a whole, organizational culture acts as a common layer shared by all. In spite of this, differing national cultures can produce differences in leadership styles, manager-subordinate relationships, and so on, especially in organizations with a high degree of multinational integration.

Because organizations have formal hierarchies of responsibility and authority, organizational culture is more amenable to carefully-planned change than are either professional or national cultures. If changes are made in a manner that is sympathetic to local national culture (as opposed to that of a distant group head office, for example), they can bring significant performance benefits. This is because organizational culture channels the effects of national and professional cultures into standard working practices.

There are many definitions of culture in the literature. The Columbia Accident Investigation Board (NASA 2003) provides a useful one for understanding culture and engineering.

Culture and Safety

Reason (1997, 191-220) describes a culture which focuses on safety as having four components:

1. A reporting culture which encourages individuals to report errors and near misses, including their own.
2. A just culture which provides *an atmosphere of trust in which people are encouraged, even rewarded, for providing essential safety-related information.*
3. A flexible culture which abandons the traditional hierarchical reporting structure in favor of more direct team-to-team communications.
4. A learning culture which is willing to draw the right conclusions from safety-related information and to implement reforms when necessary.

Weick and Sutcliffe (2001, 3) introduce the term High Reliability Organizations (HROs) (glossary). HROs have *fewer than their fair share of accidents* despite operating *under trying conditions* in domains subject to catastrophic events. Examples include *power grid dispatching centers, air traffic control systems, nuclear aircraft carriers, nuclear power generation plants, hospital emergency departments, and hostage negotiation teams.* There are five hallmarks of HROs (Weick and Sutcliffe 2001, 10):

1. **Preoccupation with Failure—**HROs eschew complacency, learn from near misses, and do not ignore errors, large or small.
2. **Reluctance to Simplify Interpretations—**HROs simplify less and see more. They “encourage skepticism towards received wisdom.”
3. **Sensitivity to Operations—**HROs strive to detect “latent failures,” defined by James Reason (1997) as systemic deficiencies that amount to accidents waiting to happen. They have well-developed situational awareness and make continuous adjustments to keep errors from accumulating and enlarging.
4. **Commitment to Resilience—**HROs keep errors small and improvise “workarounds that keep the system functioning.” They have a deep understanding of technology and constantly consider worst case scenarios in order to make corrections.
5. **Deference to Expertise—**HROs “push decision making down.” Decisions are made “on the front line.” They avoid rigid hierarchies and go directly to the person with the expertise.

The US Nuclear Regulatory Agency (2011) focuses mainly on leadership and individual authority in its policy statement on safety culture.

Historical Catastrophes and Safety Culture

The cases described in the table below are some of the many in which official reports or authoritative experts cited culture as a factor in the catastrophic failure of the systems involved.

Example	Cultural Discussion
Apollo	According to Feynman (1988), Apollo was a successful program because of its culture of *"common interest."* The *"loss of common interest"* over the next 20 years then caused *"the deterioration in cooperation, which . . . produced a calamity."*
Challenger	Vaughn (1997) states that rather than taking risks seriously, NASA simply ignored them by calling them normal—what she terms *"normalization of deviance,"* whose result was that *"flying with acceptable risks was normative in NASA culture."*
Columbia	The Columbia Accident Investigation Report (NASA 2003, 102) echoed Feynman's view and declared that NASA had a *"broken safety culture."* The board concluded that NASA had become a culture in which bureaucratic procedures took precedence over technical excellence.
Texas City - 2005	On August 3, 2005, a process accident occurred at the BP refinery in a Texas City refinery in the USA resulting in 19 deaths and more than 170 injuries. The Independent Safety Review Panel (2007) found that a corporate safety culture existed that *"has not provided effective process safety leadership and has not adequately established process safety as a core value across all its five U.S. refineries."* The report recommended *"an independent auditing function."*
The Triangle Fire	On August 11, 1911, a fire at the Triangle shirtwaist factory in New York City killed 145 people, mostly women (NYFIC 1912). The New York Factory Investigating Commission castigated the property owners for their lack of understanding of the *"human factors"* in the case and called for the establishment of standards to address this deficiency.
Nimrod	On September 2, 2006, a Nimrod British military aircraft caught fire and crashed, killing its entire crew of 14. The Haddon-Cave report (Haddon-Cave 2009) found that Royal Air Force culture had come to value staying within budget over airworthiness. Referencing the conclusions of the Columbia Accident Investigation Report, the Haddon-Cave report recommends creation of a system of detailed audits.

Relationship to Ethics

A business's culture has the potential to reinforce or undermine ethical behavior. For example, a culture that encourages open and transparent decision making and behavior makes it harder for unethical behavior to go undetected. The many differences in culture around the world are reflected in different perspectives on what is ethical behavior. This is often reflected in difficulties that international companies face when doing business globally, sometimes leading to scandals because behavior that is considered ethical in one country may be considered unethical in another. See Ethical Behavior for more information about this.

Implications for Systems Engineering

As SE increasingly seeks to work across national, ethnic, and organizational boundaries, systems engineers need to be aware of cultural issues and how they affect expectations and behavior in collaborative working environments. SEs need to present information in an order and a manner suited to the culture and personal style of the audience. This entails choices like whether to start with principles or practical examples, levels of abstraction or use cases, the big picture or the detailed view.

Sensitivity to cultural issues is a success factor in SE endeavors (Siemieniuch and Sinclair 2006).

References

Works Cited

Fasser, Y. and D. Brettner. 2002. *Management for Quality in High-Technology Enterprises.* New York, NY, USA: Wiley.

Feynman, R. 1988. "An Outsider's Inside View of the Challenger Inquiry." *Physics Today.* 41(2) (February 1988): 26-27.

Gordon, W.J.J. 1961. *Synectics: The Development of Creative Capacity.* New York, NY, USA: Harper and Row.

Haddon-Cave, C. 2009. *An Independent Review into the Broader Issues Surrounding the Loss of the RAF Nimrod MR2 Aircraft XV230 in Afganistan in 2006.* London, UK: The House of Commons.

Helmreich, R.L., and A.C. Merritt. 2000. "Safety and Error Management: The Role of Crew Resource Management." In *Aviation Resource Management,* edited by B.J. Hayward and A.R. Lowe. Aldershot, UK: Ashgate. (UTHFRP Pub250). p. 107-119.

Independent Safety Review Panel. 2007. *The Report of the BP U.S. Refineries Independent Safety Panel.* Edited by J.A. Baker. Texas City, TX, USA.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions.* Hoboken, NJ, USA: John Wiley & Sons.

Kotter, J.P. 1995. "Leading Change: Why Transformation Efforts Fail." *Harvard Business Review.* (March-April): 59-67.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications, Kings College.

NASA. 2003. *Columbia Accident Investigation Report.* Washington, DC, USA: National Aeronautics and Space Administration (NASA). August 2003.

Nuclear Regulatory Agency. 2011. "NRC Issues Final Safety Culture Policy Statement." *NRC News* (14 June 2011). Available at: http://pbadupws.nrc.gov/docs/ML1116/ML11166A058.pdf.

NYFIC. 1912. *Preliminary Report of the New York Factory Investigating Commission*. R. F. Wagner (ed). New York, NY, USA: New York Factory Investigating Commission (NYFIC).

Reason, J. 1997. *Managing the Risks of Organisational Accidents*. Aldershot, UK: Ashgate Publishing Limited.

Senge, P.M., A. Klieiner, C. Roberts, R.B. Ross, and B.J. Smith. 1994. *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization.* New York, NY, USA: Currency Doubleday.

Shields, J.L. 2006. "Organization and Culture Change." In *Enterprise Transformation,* W.B. Rouse (ed.). Hoboken, NJ, USA: John Wiley & Son.

Siemieniuch, C.E. and M.A. Sinclair. 2006. "Impact of Cultural Attributes on Decision Structures and Interfaces." Paper presented at the 11th ICCRTS Coalition Command and Control in the Networked Era. Cambridge, MA, USA. p. 1-20.

Sprenger, C. and S.T. Have. 1996. "4 Competencies of a Learning Organization." (Original title: "Kennismanagement als moter van delerende organisatie"). *Holland Management Review* Sept–Oct, p. 73–89.

Vaughn, D. 1997. *The Challenger Launch Decision: Risky Technology, Culture, and Deviance at NASA.* Chicago, IL, USA: University of Chicago Press.

Weick, K.E. and K.M. Sutcliffe. 2001. *Managing the Unexpected: Assuring High Performance in an Age of Complexity*. San Francisco, CA, USA: Jossey-Bass (Jossey-Bass acquired by Hoboken, NJ, USA: Wiley Periodicals, Inc.).

Primary References

Fasser, Y. and D. Brettner. 2002. *Management for Quality in High-Technology Enterprises.* New York, NY, USA: Wiley.

Helmreich, R.L., and A.C. Merritt. 2000. "Safety and Error Management: The Role of Crew Resource Management." In *Aviation Resource Management,* edited by B.J. Hayward and A.R. Lowe. Aldershot, UK: Ashgate. (UTHFRP Pub250). p. 107-119.

Hofstede, G. 1984. *Culture's Consequences: International Differences in Work-Related Values*. London, UK: Sage Publications.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions.* Hoboken, NJ, USA: John Wiley & Sons.

NASA. 2003. *Columbia Accident Investigation Report.* Washington, DC, USA: National Aeronautics and Space Administration (NASA). August 2003.

Reason, J. 1997. *Managing the Risks of Organisational Accidents*. Aldershot, UK: Ashgate Publishing Limited.

Senge, P.M., A. Klieiner, C. Roberts, R.B. Ross, and B.J. Smith. 1994. *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization.* New York, NY, USA: Currency Doubleday.

Additional References

Hofstede, G. 2001. *Culture's Consequences: Comparing Values, Behaviors, Institutions and Organizations Across Nations,* Second Edition. Thousand Oaks, CA, USA: Sage Publications.

Hofstede, G. 2010. *Cultures and Organizations: Software for the Mind,* Third Edition. New York, NY, USA: McGraw Hill.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Enabling Teams

Enabling Teams

This knowledge area focuses on enabling a team to perform SE. Once that is done using the techniques described here, the knowledge found in Part 3, Systems Engineering and Management, about how to perform SE can be applied. Part 5, Enabling Systems Engineering, to which this knowledge area belongs, explores how systems engineering (SE) is enabled at three levels of organization: the business or enterprise, the team, and the individual.

For the sake of brevity, the term "business" is used to mean "business or enterprise" throughout most of this knowledge area. For a nuanced explanation of what distinguishes a business from a enterprise, see Enabling Systems Engineering.

Topics

Each part of the SEBoK is composed of knowledge areas (KAs). Each KA groups topics together around a theme related to the overall subject of the part. This KA contains the following topics:

- Team Capability
- Team Dynamics
- Technical Leadership in Systems Engineering

Overview

Products, enterprise systems, and services are developed, delivered, and sustained with the contributions of systems engineers, who also coordinate the technical aspects of the multiple projects that comprise a program. These activities require certain individuals to work in a cooperative manner to achieve shared objectives based on a common vision—that is, as teams. Not every group of individuals working together is a team. To perform SE activities efficiently and effectively, the capabilities of and dynamics within the team must be specifically attuned to SE.

Although individuals sometimes perform SE activities, it is more usual to find project teams performing SE activities while providing specialty engineering capabilities (see Systems Engineering and Specialty Engineering). Not all who perform SE activities are labeled "systems engineers." Thus, electrical, mechanical, and software engineers, service providers, or enterprise architects in IT organizations may lead or be members of teams that perform SE tasks. Those individuals are referred to as systems engineers in this knowledge area, regardless of their job titles within their organizations.

This knowledge area is concerned with methods, tools, and techniques for enabling project teams to perform SE activities. Its first topic, Team Capability, answers the questions

- How do businesses determine value added by SE activities performed by project teams?
- How does an organization determine the efficiency and effectiveness of SE activities performed by project teams?

Its other topic, Team Dynamics, answers the question

- How are group dynamics crucial to enabling systems engineers to perform work and achieve goals?

Topics from elsewhere in the SEBoK that cover related questions include Relationships between Systems Engineering and Project Management and The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships, which answer the question

- What do managers need to know about managing systems engineers and project teams that perform SE activities?

References

Works Cited

None.

Primary References

Brooks, F. 1995. *The Mythical Man-Month,* Anniversary Edition. Reading, MA, USA: Addison Wesley.

Curtis, B., W.E. Hefley, and S.A. Miller. 2001. *People Capability Maturity Model (P-CMM),* Version 2.0. Pittsburg, PA, USA: Software Engineering Institute (SEI). CMU/SEI-2001-MM-01. Accessed on June 8, 2012. Available at http://www.sei.cmu.edu/library/abstracts/reports/01mm001.cfm.

DeMarco, T. and T. Lister. 1999. *Peopleware: Productive Projects and Teams,* 2nd ed. New York, NY, USA: Dorset House.

Eisner, H. 2008.*Essentials of Project and Systems Engineering Management*, 3rd ed. Hoboken, NJ, USA: John Wiley and Sons.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, NJ, USA: John Wiley & Sons.

Forsyth, D.R. 2010. *Group Dynamics,* 5th edition. Belmont, CA, USA: Wadsworth, Cengage Learning.

Hase, S. 2000. "Measuring Organisational Capability: Beyond Competence", Paper presented at Future Research, Research Futures: Australian Vocational Education and Training Research Association (AVETRA) Conference (2000). Accessed on June 8, 2012. Available at http://www.avetra.org.au/abstracts_and_papers_2000/shase_full.pdf.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205*. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2010-003.

NASA. 2011. *Academy of Program/Project and Engineering Leadership (APPEL), NASA APPEL Performance Enhancement.* Accessed on September 15, 2011. Available at http:/ / www. nasa. gov/ offices/ oce/ appel/ performance/index.html.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Team Capability

The capability of a team to perform systems engineering (SE) depends on having competent personnel, adequate time, sufficient resources and equipment, and appropriate policies and procedures (Torres and Fairbanks 1996).

The team should have a charter. Staff must be proficient in the needed competencies and must work together with the right attitude, under the right organization, and with appropriate tools, training, and processes such as configuration management and peer review.

Those responsible for the team attaining the desired capability need to organize, staff, develop, and assess the team. Techniques for pilot projects, post-mortem analysis, and lessons learned can be applied as well.

Organizing the Team

Project teams, and the roles of systems engineers within those teams, depend on factors such as the nature, size, and scope of the project, the organization's preferred way of organizing teams, and external constraints such as a larger program in which the project may be embedded. Options range from a dedicated team of systems engineers, to Integrated Product Teams, to teams that include other kinds of engineers that perform systems engineering.

Systems engineers and SE teams may play the roles of technical leads, consultants, or advisers; this also influences the ways in which SE teams are organized. In some organizations, systems engineers and SE teams provide technical leadership; they perform requirements analysis and architectural design, conduct trade studies, and allocate requirements and interfaces to the various elements of a system. In addition, they work with component specialists, develop integration plans and perform system integration, verification, and validation. Depending on the scope of effort, they may also install the system and train the operators and users; provide ongoing services to sustain the system; and retire/replace an aged system. Systems engineers may be housed within a functional unit of an organization and assigned, in matrix fashion, to projects and programs, or they may be permanently attached to a project or program for the duration of that endeavor. They may be organized based, in part, on their domain of expertise, such as finance or telecommunications. For additional information on organizational options see Determining Needed Systems Engineering Capabilities in Businesses and Enterprises.

In other cases, one or more systems engineers may provide consulting or advisory services, as requested, to projects and programs. These engineers may be dispatched from a central pool within an organization, or they may be hired from an outside agency.

An SE team can be organized by job specialization, where each SE team member (or each SE sub-team) plays a different role; for example, requirements engineering, system architecture, integration, verification and validation, field test, and installation and training; in this case the various job specializations are typically coordinated by a lead systems engineer.

Alternatively, an SE team can be organized by subsystem where each SE team member (or SE sub-team) performs the previously indicated functions for each of the subsystems with a top-level team to coordinate requirements allocation, interfaces, system integration, and system verification and validation.

Ideally, roles, responsibilities, and authority will be established for each project or program and used to determine the optimal way to organize the team. Sometimes, however, an *a priori* organizational, project, or program structure may determine the structure, roles, responsibilities, and authority of the SE team within a project or program; this may or may not be optimal.

Within a project, a systems engineer or SE team may occupy a staff position subordinate to the project manager, as indicated in Figure 1 or conversely, the SE team may provide the authoritative interface to the customer with the project manager or management team, serving in a staff capacity, as indicated in Figure 2. In both cases, SE and project management must work synergistically to achieve a balance among product attributes, schedule, and budget. Eisner (2008) lays out various approaches to organizing systems engineers. For additional information see Systems

Engineering and Project Management.

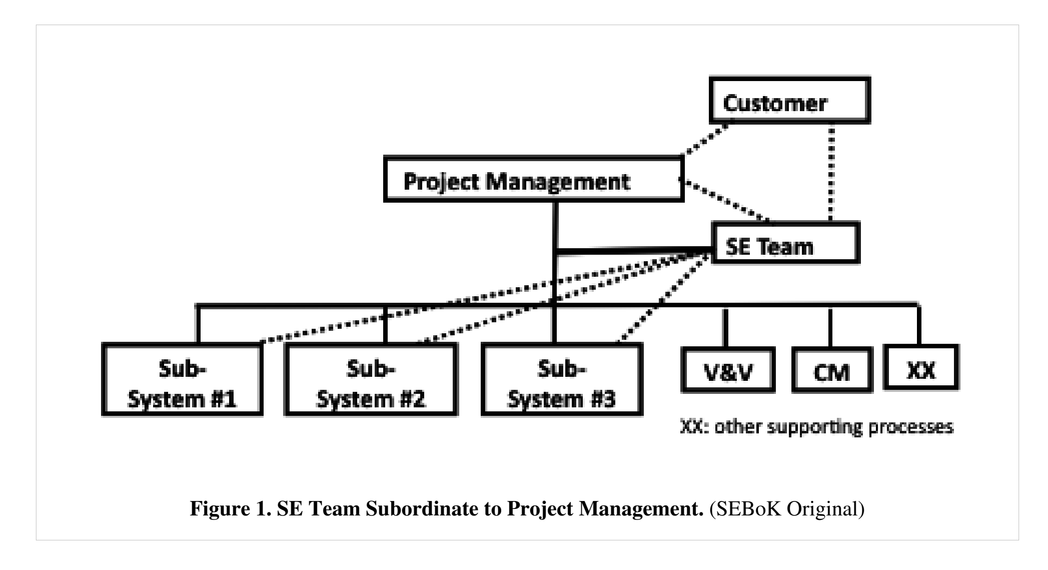


Figure 1. SE Team Subordinate to Project Management. (SEBoK Original)

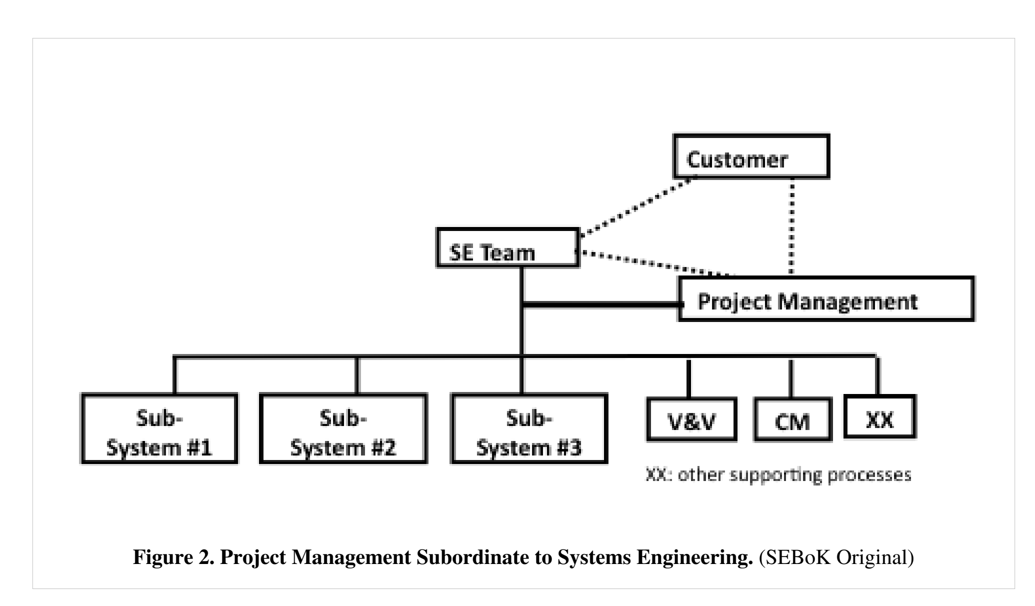


Figure 2. Project Management Subordinate to Systems Engineering. (SEBoK Original)

In scaling up to the program level, the considerations portrayed in Figures 1 and 2 can be generalized so that a top-level SE team provides coordination among the subordinate projects. In this case, each project has an SE team, and within each project the SE team members can be organized in either of the ways indicated in the figures. When scaling up to programs, each of the sub-systems in Figures 1 and 2 are separate, coordinated projects.

The models presented in Figures 1 and 2 can be scaled down to smaller projects, where an individual systems engineer performs the SE activities, either in the subordinate position of Figure 1 or the superior position of Figure 2. In this case, there is a single subsystem (i.e., the system) and the supporting functions may be provided by the systems engineer or by supporting elements of the larger organization.

The roles to be played by members of a SE team are influenced by the structures adopted as part of the organizational strategy of the business in which the team is operating (see Systems Engineering Organizational Strategy). In Product Centered Organizations, for example, an Integrated Product Team (IPT) is assigned to each element of the system breakdown structure (SBS). Each IPT consists of members of the technical disciplines necessary to perform systems engineering functions for that element of the system.

At the program level there is a top-level IPT commonly called a SE and integration team (SEIT), whose purpose is to oversee all of the lower level IPTs. Some specialists, such as reliability and safety engineers, may be assigned to a team to cover all elements within a given level of the SBS. These teams are sometimes called Analysis and

Integration teams (AITs), and are created at various levels of the SBS as needed.

Organizing communication and coordination among a group of systems engineers should follow the well known *7 ± 2* rule because the number of communication paths among *N* engineers is *N(N-1)/2*; i.e., the number of links in a fully connected graph (Brooks 1995). There are 10 communication paths among 5 engineers, 21 among 7 engineers, and 36 among 9 engineers. An SE team of more than 10 members (45 paths) should be organized hierarchically with a top-level team leader. Sub-teams can be partitioned by product subsystem or by process work activities (analysis, design, integration).

Staffing the Team

Once the organizational structure of the SE team is understood, the team can be staffed. As noted in Enabling Individuals, competency of an individual is manifest in the knowledge, skills, abilities, and attitudes needed for the individual to perform a specific task efficiently and effectively. Different levels of competency may be needed in different situations. Competencies include occupational competence, social competence, and communication competence. Competent systems engineers, for example, have SE knowledge, skills, and ability; engage in systems thinking; possess emotional intelligence; and have good communication and negotiation skills. In addition, competent systems engineers are typically competent within specific domains (e.g. aerospace, medicine, information technology) and within specific process areas of systems engineering (e.g., requirements, design, verification and validation). (See Part 3, Systems Engineering and Management for more information on specific process areas.) The article on Roles and Competencies includes a summary of SE competency models. Based on the context, these competency models are tailored to match the needs of each project. The roles within the team are defined, and competencies are linked to the roles. The lists of competencies given in those models are most often distributed among the members of a SE team. It is not often that a single individual will possess the full list of competencies given in these models.

In addition to individual competencies to perform SE roles, the collective SE competencies needed by a team depend on additional factors including the domain, the stakeholders, the scope of the effort, criticality of outcome, new initiative versus enhancement, and the responsibilities and authority assigned to the team. For example, collective SE competencies needed to develop the IT enterprise architecture for a small company are quite different from those needed to develop the architecture of an aircraft which is engineered and manufactured in a distributed fashion around the world.

To determine the collective set of competencies an SE team needs to conduct a project or program, perform the following steps:

1. Identify the context, to include
   1. domain
   2. stakeholders
   3. organizational culture
   4. scope of effort
   5. criticality of the product, enterprise endeavor, or service
   6. new initiative or sustainment project
2. Clarify the responsibilities, authority, and communication channels of the systems engineering team
3. Establish the roles to be played by systems engineers, and other project personnel as determined by context, responsibilities, and authority
4. Determine the required competencies and competency levels needed to fill each of the systems engineering roles
5. Determine the number of systems engineers needed to provide the competencies and competency levels for each role
6. Determine the availability of needed systems engineers
7. Make adjustments based on unavailability of needed systems engineers

8. Organize the systems engineering team in a manner that facilitates communication and coordination within the SE team and throughout the project or program
9. Consult stakeholders to ask "What are we missing?"

Competency models and skills inventories, such as INCOSE (2010) and Curtis et al. (2001), can be used as checklists to assist in determining the needed competencies and competency levels for a product, enterprise, or service. (See Roles and Competencies.)

When the needed competencies, competency levels, and capacities have been determined, one of two situations will arise: optimally, the number of systems engineers who have the needed competencies and competency levels to fill the identified roles will be available; or, they will either be unavailable or cannot be provided because of insufficient funding. For example, a new initiative may need a lead engineer, a requirements engineer, a systems architect and a systems integrator-tester to accomplish systems engineering tasks. Budgetary constraints may indicate that only two of the four roles can be supported. Compromises must be made; perhaps the system architect will be the lead engineer and the requirements engineer will also be assigned the tasks of system integration and testing even though he or she does not have the desired skill and experience (i.e., competency level) in integration and testing.

Developing the Team

Before a team that performs SE can be effective, it needs to establish its own identity, norms, and culture. The well-known four stages of *"forming, storming, norming, performing"* (Tuckman 1965, 384-399) indicate that a SE team needs time to form, for the members to get to know and understand each other as well as the tasks to be performed, and to work out how best to work together. It is also important that care is taken to ensure, to the extent possible, assignment of roles and responsibilities that would allow SE team members to satisfy their individual goals (Fraser 2010).

The *cost and time to cohesion* can be minimized by good selection and management of the SE team, consistent training across the business so that team members have a common framework of understanding and language for their work, good "infostructure" to allow easy and useful sharing of information, and shared behavioral norms and values. Conversely, in cross-site, inter-company and international SE teams, more time must be allowed for team formation. SE teams are more effective if attention is given to ensuring that each member's work satisfies their individual goals as well as the team and organizational objectives (Fraser 2010).

According to Stephenson and Weil (1992), capable people are:

> *those who know how to learn; are creative; have a high degree of self-efficacy, can apply competencies in novel as well as familiar situations; and work well with others. In comparison to competency, which involves the acquisition of knowledge and skills, capability is a holistic attribute.*

The results of a survey by Steward Hase (2000) concluded that the following are significant contributors to the human elements of capability:

- Competent People
- Working in Teams
- Visible Vision and Values
- Ensuring Learning Takes Place
- Managing the Complexity of Change
- Demonstrating the Human Aspects of Leadership
- Performing as Change Agents
- Involving People in Change
- Developing Management Talent
- Committing to Organizational Development

These attributes of human capability apply to all members of an organization, including systems engineers, both as individuals and as members of project teams.

DeMarco and Lister (1999) discuss “teamicide” techniques by which management, perhaps unintentionally, practices *sure fire techniques to kill teams*. Teamicide techniques include

- physical separation of team members
- fragmentation of time
- unrealistic schedules
- excessive overtime

Methods for developing and improving SE capabilities within teams include building cohesive teams, conducting pilot projects, participating in and studying post-mortem analyses, and preparation and examination of lessons learned. Members of a cohesive systems engineering team have a strong sense of commitment to the work and to the other team members. Commitment creates synergy, which results in performance greater than the sum of the performance of the individual team members.

Some key indicators of a cohesive systems engineering team (Fairley 2009, 411) are

- clear understanding of systems engineering roles and responsibilities
- shared ownership of systems engineering work products
- willingness of systems engineers to help one another and to help other project members
- good communication channels among systems engineers and with other project elements
- enjoyment of working together

Negations of these indicators—the hallmarks of a dysfunctional team—are

- confusion of systems engineering roles and responsibilities
- protective ownership of systems engineering work products
- unwillingness to help one another
- absence of good communications among systems engineers and with other project elements
- personal dislike of one or more other systems engineering team members

Techniques for building and maintaining cohesive systems engineering teams include

- an appropriate number of systems engineering team members
- a correct mix of systems engineering competencies
- celebration of project milestones
- team participation in off-site events
- social events that include family members

Assessing the Team

Performance evaluation is most often conducted for individuals. Robbins (1998, 576) states the historic belief that individuals are the core building blocks around which organizations are built. However, it is also important to assess the team's capability and performance. To design a system that supports and improves the performance of teams, including SE teams, Robbins offers four suggestions:

1. Tie the SE team's performance and the overall project team's results to the organization's goals
2. Begin with the team's customer (glossary) and the work process the team follows to satisfy customer's needs
3. Measure both team and individual performance and compare them to organizational norms and benchmarks
4. Train the team to create its own measures.

Robbins' approach can be applied in the context of SE:

1. Tie the SE and overall project team's results to the project's and the organization's goals. Use measures that apply to goals the team must achieve. For SE in particular, the team effort should be tied to the product or service which

the organization seeks to deliver. The end product for the SE team should not be only the SE work products but the delivered products and services provided by the project. For more information on general SE assessment, see Systems Engineering Assessment and Control.

2. Consider the SE team's customers and more broadly the key stakeholders and the work processes that the SE team follows to satisfy customer needs. SE customers and stakeholders can be internal or external; the internal customers of systems engineering are the other project elements that depend on systems engineering work products and services, which can be evaluated for on-time delivery of quantity and quality. The process steps can be evaluated for waste and cycle time; i.e., efficiency and effectiveness.
3. Assess both individual and team performance. Define the roles of each SE team member in terms of the tasks that must be accomplished to produce the team's work products. For more information on individual assessment, see Assessing Individuals.
4. Finally, have the team define its own measures of achievement of goals. This helps all members of the team to understand their roles, while also building team cohesion.

As an example, NASA's Academy of Program/Project and Engineering Leadership (APPEL) provides a service where team performance is assessed and interventions are provided to the team for specific gaps in performance (NASA 2011). This performance enhancement service increase a project's probability of success by delivering the right support to a project team at the right time. APPEL offers the following assessments:

- Project/Team Effectiveness — Measures effectiveness of a team’s behavioral norms
- Individual Effectiveness — Measures effectiveness of an individual’s behavioral norms
- Project/Team Process Utilization — Measures the extent of a team’s utilization of key processes
- Project/Team Knowledge — Covers topics that NASA project personnel should know in order to perform in their jobs

The APPEL approach can be applied to assessing the performance of a SE team and individual systems engineers.

Further Techniques for Building Team Capability

Further techniques for developing SE capabilities within teams include conducting pilot projects, preparing post-mortem analyses, and participating in and studying lessons learned.

Pilot Projects

Pilot projects are an effective mechanism by which SE teams can build team cohesion, acquire new skills, and practice applying newly acquired skills to projects and programs. Pilot projects can be conducted for the sole purpose of skills acquisition, or additionally they can be conducted to determine the feasibility of a proposed approach to solving a problem. Feasibility studies and acquisition of new team skills can be combined in proof-of-concept studies. Primary inhibitors to conducting SE pilot projects are the time required and diversion of personnel resources.

Post-mortem Analysis

A post-mortem analysis identifies areas for improvement of SE performance in future projects and programs. Inputs to a post-mortem analysis include

- personal reflections and recollections of project personnel and other stakeholders;
- email messages, memos, and other forms of communication collected during a project or program;
- successful and unsuccessful risk mitigation actions taken; and
- trends and issues in change requests and defect reports processed by the change control board.

Team participation in a post-mortem analysis allows SE team members to reflect on past efforts, which can lead to improved team capabilities for future projects or, if the present team is being disbanded, improved individual ability to participate in future systems engineering teams.

Inhibitors for effective post-mortem analysis include not allocating time to conduct the analysis, failure to effectively capture lessons-learned, failure to adequately document results, reluctance of personnel to be candid about the performance of other personnel, and negative social and political aspects of a project or program. Mechanisms to conduct effective post-mortem analyses of SE projects include using a third party facilitator, brainstorming, Strength-Weakness-Opportunity-Threat (SWOT) analysis, fishbone (Ishikawa) diagrams, and mind mapping.

Lessons Learned

Lessons learned in SE include both positive and negative lessons. Experiences gained and documented from past projects and programs can be an effective mechanism for developing and improving the capabilities of a team that performs SE tasks. Studying past lessons learned can aid in team formation during the initiation phase of a new project. Lessons learned during the present project or program can result in improved capabilities for the remainder of the present project and for future projects. Inputs for developing and documenting SE lessons learned include results of past post-mortem analyses plus personal recollections of the team members, informal *war stories*, and analysis of email messages, status reports, and risk management outcomes. Inhibitors for developing and using SE lessons learned include failure to study lessons learned from past projects and programs during the initiation phase of a project, failure to allocate time and resources to developing and documenting lessons learned from the present project or program, and reluctance to discuss problems and issues.

References

Works Cited

Brooks, F. 1995. *The Mythical Man-Month*, anniversary edition. Reading, MA, USA: Addison Wesley.

Curtis, B., W.E. Hefley, and S.A. Miller. 2001. *People Capability Maturity Model (P-CMM)*, version 2.0. Pittsburgh, PA, USA: Software Engineering Institute (SEI). CMU/SEI-2001-MM-01. Accessed April 24, 2013. Available: http://www.sei.cmu.edu/library/abstracts/reports/01mm001.cfm.

DeMarco, T., and T. Lister. 1999. *Peopleware: Productive Projects and Teams*, 2nd ed. New York, NY, USA: Dorset House.

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons.

Fairley, R.E. 2009. *Managing and Leading Software Projects*. Hoboken, NJ, USA: John Wiley & Sons.

Fraser, D. 2010. *Relationships Made Easy: How to Get on with The People You Need to Get on with...and Stay Friends with Everyone Else*. Worchestershire, UK: Hothive Books.

Hase, S. 2000. "Measuring Organisational Capability: Beyond Competence." Paper presented at Future Research, Research Futures: Australian Vocational Education and Training Research Association (AVETRA) Conference

(2000). Accessed September 14, 2011. Available: http:/ / www. avetra. org. au/ abstracts_and_papers_2000/ shase_full.pdf.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205*. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2010-003.

NASA. 2011. *Academy of Program/Project and Engineering Leadership (APPEL), NASA APPEL Performance Enhancement*. Accessed September 15, 2011. Available: http://appel.nasa.gov/team/request-support/.

Robbins, S.P. 1998. *Organizational Behavior: Concepts, Controversies, Applications*, 8th ed. Upper Saddle River, NJ, USA: Prentice Hall. p. 576.

Stephenson, J. and S. Weil. 1992. *Quality in Learning: A Capability Approach in Higher Education*. London, UK: Kogan Page.

Torres, C., and D. Fairbanks. 1996. *Teambuilding: The ASTD Trainer's Sourcebook*. New York, NY, USA: McGraw-Hill.

Tuckman, B. 1965. "Developmental Sequence in Small Groups." *Psychological Bulletin*. 63 (6): 384-99.

Primary References

Brooks, F. 1995. *The Mythical Man-Month*. Anniversary Edition. Reading, MA, USA: Addison Wesley.

Curtis, B., W.E. Hefley, and S.A. Miller. 2001. *People Capability Maturity Model (P-CMM)*, Version 2.0. Pittsburg, PA, USA: Software Engineering Institute (SEI). CMU/SEI-2001-MM-01. Accessed on June 8, 2012. Available at http://www.sei.cmu.edu/library/abstracts/reports/01mm001.cfm.

DeMarco, T. and T. Lister. 1999. *Peopleware: Productive Projects and Teams*. 2nd ed. New York, NY, USA: Dorset House.

Eisner, H. 2008. *Essentials of Project and Systems Engineering Management*. 3rd ed. Hoboken, NJ, USA: John Wiley & Sons.

Fairley, R.E. 2009. *Managing and Leading Software Projects*. Hoboken, NJ, USA: John Wiley & Sons.

Hase, S. 2000. "Measuring Organisational Capability: Beyond Competence". Paper presented at Future Research, Research Futures: Australian Vocational Education and Training Research Association (AVETRA) Conference (2000). Accessed on June 8, 2012. Available at http://www.avetra.org.au/abstracts_and_papers_2000/shase_full.pdf.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205*. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2010-003.

NASA. 2011. *Academy of Program/Project and Engineering Leadership (APPEL), NASA APPEL Performance Enhancement.* Accessed on May 2, 2014. Available at http://appel.nasa.gov/team/request-support/.

Torres, C. and D. Fairbanks. 1996. *Teambuilding: The ASTD Trainer's Sourcebook*. New York, NY, USA: McGraw-Hill.

Additional References

Fasser, T. and D. Brettner. 2002. *Management for Quality in High Technology Enterprises.* New York, NY, USA: Wiley.

INEEL 2004. *A Project Management and Systems Engineering Structure for a Generation IV Very High Temperature Reactor.* Idaho Falls, ID, USA: Idaho National Engineering and Environmental Laboratory, NEEL/CON-04-02175. Accessed on September 14, 2011. Available at http://www.inl.gov/technicalpublications/Documents/2808490.pdf.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Team Dynamics

A systems engineering (SE) team (glossary) is a group of individuals who cooperatively perform a collection of SE tasks based on a shared vision and a common set of engineering objectives. Applying the practical considerations of group dynamics is essential to enabling SE teams to successfully perform SE activities. The interplay of the behaviors of humans in groups is varied, changing, and inescapable. Nevertheless, study of these behaviors has yielded valuable insight and knowledge on the dynamics of individuals within groups. The awareness and application of group dynamics is crucial to facilitating systems engineers' performance of work and achievement of their goals.

The study of group dynamics was initially within the province of psychology and later within sociology. The importance of group dynamics to successful teams has led other disciplines such as business management to study and apply team dynamics.

History

The origins of the study of group dynamics began with Gustave Le Bon. Le Bon wrote *La psychologie des fouls* in 1895, which was translated into English as *The Crowd: A Study of the Popular Mind* a year later. Sigmund Freud wrote *Group Psychology and the Analysis of the Ego* in 1922 responding to Le Bon's work. Kurt Lewin is acknowledged as the "founder of social psychology", coining the term **group dynamics**. He founded the Research Center for Group Dynamics at the Massachusetts Institute of Technology in 1945, relocating in 1948 to the University of Michigan. Wilfred Bion studied group dynamics from a psychoanalytical perspective. He help found the Tavistock Institute of Human Relations in 1947. In that same year, both the Research Center for Group Dynamics and the Tavistock Institute of Human Relations founded the journal *Human Relations*. The study of group dynamics is now worldwide, active, and well established.

Nature of Groups

Groups are endemic to human existence and experience; humans are by nature social animals. Consequentially, an informed understanding of the nature of groups is very useful in enabling teams to perform SE. Research into group behavior reveals that the nature of a group can be described by interaction, goals, interdependence, structure, unity, and stage. (Forsyth 2010, 5-10)

Interaction

Communication (both verbal and non-verbal) among members within a group produces constantly changing and varied interactions. Group dynamics are more than the sum of the interactions between individual members; group interactions create synergistic behaviors and results. Interactions can be placed into two categories (1) socio-emotional interactions and (2) task interactions (Bales 1950, 1999).

Goals

All groups exist for the purpose of achieving one or more goals. The goals provide the basis for the group’s tasks. The tasks accomplished by the group can be categorized into activities and characterized by a Circumplex Model (McGrath 1984, 61), which establishes four quadrants, where the X-axis is *choose* vs. *execute* and the Y-axis is *generate* vs. *negotiate*.

Interdependence

Interdependence is *the state of being dependent to some degree on other people, as when one’s outcomes, actions, thoughts, feelings, and experiences are determined in whole or in part by others*. Interdependence can be categorized into five types (1) mutual, reciprocal; (2) unilateral; (3) reciprocal, unequal; (4) serial; and (5) multi-level. (Forsyth 2010, 8)

Structure

Structure includes the organization and patterned behaviors of a group. Structure can be deliberately devised and/or emergently observed. Most groups have both kinds of structures, which are evinced in the roles and norms of the group. *The roles of leader and follower are fundamental ones in many groups, but other roles — information seeker, information giver, elaborator, procedural technician, encourager, compromiser, harmonizer — may emerge in any group* (Benne and Sheats 1948; Forsyth 2010, 9). Norms are the rules that govern the actions of group members; norms can include both formal and informal rules.

Cohesion

The *interpersonal forces that bind the members together in a single unit with boundaries that mark who is in the group and who is outside of it* constitute a group’s cohesion (Dion 2000). Cohesion is an essential quality of group; it can vary from weak to strong. A team cannot perform effectively without strong group cohesion.

Stage

Groups exhibit stages of development. Being comprised of people, it is not surprising that groups collectively demonstrate the dynamics and growth of the individuals that constitute the group members. The most well-known and wide-spread model of the stages of group development was developed by Bruce Tuckman. The initial model identified the sequence of group development as (1) Forming, (2) Storming, (3) Norming, and (4) Performing (Tuckman 1965). He later added a final stage to the model: (5) Adjourning (Tuckman and Jensen 1977). While Tuckman’s model is sequential, others have observed that groups actually may recursively and iteratively progress through the different stages (Forsyth 2010, 20).

Practical Considerations

The dynamics associated with creating, nurturing, and leading a team that will successfully achieve the team's goals is important and challenging. Although psychologists and sociologists have conducted and continue to conduct research to understand team dynamics, the profession of business management has additionally sought to develop practical guidance for utilizing and applying this knowledge to foster high-performance teams. Accordingly, business management has focused its contribution to the field of team dynamics by publishing practical guidebooks to analyze the problems and focus on developing solutions to the problems of team dynamics (see Additional References). There are many consultancy firms throughout the world that assist organizations with the application of practical knowledge on team dynamics. Successful systems engineering teams would do well to not ignore, but rather take advantage of this knowledge.

References

Works Cited

Bales, R.F. 1950. *Interaction Process Analysis: A Method for The Study of Small Groups.* Reading, MA, USA: Addison-Wesley.

Bales, R.F. 1999. *Social Interaction Systems: Theory and Measurement.* New Brunswick, NJ, USA: Transaction.

Benne, K.D. and P. Sheats. 1948. "Functional Roles of Group Members." *Journal of Social Issues.* 4 (2): 41-49. Blackwell Publishing Ltd.

Dion, K.L. 2000. "Group Cohesion: From 'Field of Forces' to Multidimensional Construct." *Group Dynamics: Theory, Research, and Practice.* 4 (1): 7-26. Washington DC, USA: American Psychological Association.

Forsyth, D.R. 2010. *Group Dynamics,* 5th edition. Belmont, CA, USA: Wadsworth, Cengage Learning.

McGrath, J.E. 1984. *Groups: Interaction and Performance.* Upper Saddle River, NJ, USA: Prentice Hall.

Tuckman, B.W. 1965. "Developmental Sequence in Small Groups." *Psychological Bulletin.* 63 (6):384-399. Washington DC, USA: American Psychological Association.

Tuckman, B.W. and M.C. Jensen. 1977. "Stages of Small Group Development Revisited." *Group and Organization Management* 2 (4): 419-427. Thousand Oaks, CA, USA: Sage Publications.

Primary References

Forsyth, D.R. 2010. *Group Dynamics,* 5th edition. Belmont, CA, USA: Wadsworth, Cengage Learning.

Additional References

Scholtes, P.R., B.L. Joiner, and B.J. Streibel. 2003. *The Team Handbook,* 3rd edition. Edison, NJ, USA: Oriel Inc.

Larson, C.E. and F.M.J. LaFaso. 1989. *Teamwork: What Must Go Right, What Can Go Wrong.* Newbury Park, CA, USA: Sage Publications, Inc.

Lencioni, P. 2002. *The Five Dysfunctions of a Team: A Leadership Fable*. San Francisco, CA, USA: Jossey-Bass.

Lencioni, P. 2005. *Overcoming the Five Dysfunctions of a Team.* San Francisco, CA, USA: Jossey-Bass.

McShane, S.L. and M.A. Von Glinow. 2010. *Organizational Behavior: Emerging Knowledge and Practice for the Real World.* New York, NY, USA: McGraw-Hill/Irwin.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Technical Leadership in Systems Engineering

Leadership is an important but often overlooked component of technical projects and programmes. It addresses the performance of people: their behaviours, their ability to think individually and collectively, their motivation and energy. Technical leadership in systems engineering creates the environmental conditions conducive to good performance, supporting shared understanding, innovation, problem solving, resilience and learning. Leadership is thus complementary to management, which directs specific activities to deliver outputs. A systems engineering leader may lead a team of systems engineers for a project or program, or may be the only systems engineer in a team of diverse members involved in project or program (e.g. other engineers, IT personnel, service providers). There are various models and styles of leadership and key to success is matching leadership to the needs of a situation. 'Models' of leadership describe the mechanisms by which leadership arises and operates (e.g. situationally-driven or caused by a charismatic individual). 'Styles' of leadership describe the manner in which a leader (or a leadership team) leads (e.g. task-focussed or people-focussed; autocratic, democratic or "laissez-faire" (Lewin et al., 1939)).

There is a vast amount of literature addressing leadership issues from multiple points of view, including philosophical, psychological and emotional considerations (Yukl, 2012). This article highlights key aspects of leadership theory to help systems engineers understand how they may influence the success of their team and organisation. This provides the basic building blocks for adapting leadership behaviours at work. The pragmatic aspects of leading team members involved in systems engineering are summarised in section 1.11. This section highlights the need to use different approaches to leadership across the systems engineering context, and it is therefore important be able to understand and adopt the leadership behaviours discussed in the preceding sections, as judged appropriate. Related knowledge areas and articles are in Part 5 Enabling Systems Engineering and the Part 6 Knowledge area Systems Engineering and Project Management.

Attributes of Effective Leaders

Traditional attitudes to technical leadership

The need for leadership in an engineering environment has not been widely emphasized or understood. Traditional academic engineering curricula do not cover the development of leadership skills, and industry professionals tend be task-oriented, with project leaders perceived in terms of power and authority (Toor and Ofori, 2008). In many cases, technical organisations focus on management rather than leadership. "Managers are people who do things right while leaders are people who do the right thing" (Bennis and Nanus, 1985). Doing the right thing is not only about identifying the right approach in the first place; it is also about taking responsibility for understanding and challenging the progression of a project or programme in a continuous manner. It is now recognised that leadership is a critical component of successful projects and programmes, and that technical leadership is likely to be a distributed responsibility.

Great Man theories: traits and charisma models of leadership

Early concepts of leadership were driven by views of leaders as heroic figures, with particular qualities that made them different to other people. The notion of 'charisma' was used to describe the ability to charm and influence followers. Numerous studies have been conducted to try and define the particular personality traits that made someone a born leader. The findings are not clear-cut, partly because there are many different models of personality which produce different results (Hippocrates first identified 4 personality dimensions in the 5th Century BC , and many different conceptualisations have been devised since then). Personality tests should be used with great caution because each test has been developed for specific purposes and contexts, and is only valid within those parameters. For tests to be valid they must undergo a strict set of tests with extensive data sets, and then they must be used exactly as specified by the validation process. The current best consensus of evidence is that there are 5 main

dimensions of personality: Extraversion (talkative, sociable); Agreeableness (good natured, co-operative); Conscientiousness (responsible, tidy); Neuroticism (general level of anxiety or composure); and Openness (to new experiences). This 5-Factor model has good validity across literate populations, but even this model may not be universal (Gurven at al., 2013). There is some evidence that extraversion is associated with leadership roles, but this is not always a predictor of success, and may reflect a cultural stereotype which leads to people who behave like leaders being more likely to get leadership roles. Different contexts will change the value of extraversion (and other traits) in a leader. (See Judge et al. (2002) for a meta-analysis of the literature).

In business settings, the Myers-Briggs Type Indicator (MBTI) personality test is often used as part of guided discussions to assist with self-development (although it lacks the important personality dimension of 'neuroticism'). People can use their MBTI profiles to help them use their strengths more effectively. Occasionally, MBTI is misused as a basis for selection, especially for leadership roles. The evidence indicates this is not justified (National Research Council, 1991).

Transactional and Transformational Leadership Styles

Certain behaviours have been associated with successful leadership. These behaviours arise from the style of leadership and in particular the attention paid to the task compared to team relationships. Such differences are described as transactional and transformational styles (Burns, 1978). Transactional leadership is closely allied to management, focussed on defined task outputs and incentivising people to follow directions by rewarding and punishing.

Transformational leadership is concerned with achieving outcomes through the development of the people (team building), building trust, developing a shared vision, motivation, cultivating relationships and sharing knowledge. Both types of leadership have value, but transformational leadership is needed for developing the culture of an organisation, for ensuring qualities such as safety, adaptability, learning and improvement, and is usually considered the most valuable form of leadership.

Understanding that different styles have value for different situations provides the basis for leadership models that recognise the interactions between style and situation. Fiedler's Contingency Model, Hersey et al.'s Situational Model and House's Path-Goal Theory provide useful variations on this approach (all described below).

Contingency Model of leadership

Fielder's Contingency Model (1964) states that there is no one best style of leadership. Effectiveness is about the match between leadership style (defined as task or relationship-oriented) and situation (defined by: the degree to which the leader is supported by the group; the degree to which the task is clearly structured; and the degree to which the leader can reward and punish team members). Fiedler devised a way of assessing leaders' styles by measuring their attitude to their 'least preferred co-worker' or LPC. In general terms, leaders who are more negative about their LPC are task-oriented and focus on organising. Leaders who are more positive towards their LPCs are more able to avoid conflict, promote innovation and learning and are better at making complex decisions. In moderate situations (not extreme in any of the three situation dimensions), the more positive, relationship oriented leaders appear to more successful (Valle & Avella, 2003). This contingency model of leadership was found to predict leadership style in an information systems engineering environment, where leadership functions were distributed across technical experts and the end-user (Franz, 1985).

Situational Theory

Situational Theory offers a model of leadership in which any individual leader adapts his or her style according to the needs of the situation. For example, they can learn to change from being task-focussed to being relationship focussed. They may also adapt according to their own changing status. Hersey, Blanchard and Johnson (2001) describe four modes that leaders can adapt between, according to the nature of the members of the team or organisation: delegating, supporting, coaching, and directing. In this situational model, leadership is a learned skill based on understanding context and self-awareness.

Path-Goal Model

The Path-Goal Theory describes the leader's role as helping followers to develop behaviours that allow them to achieve the goals (House and Mitchell, 1974). Leaders are facilitators for others' achievements, for example providing resources, associations, knowledge and support. Leaders are members of a community of practice united in a common enterprise and sharing a common culture: history, values, ways of doing things, ways of talking (Drath and Palus, 1994). In technical leadership, this means helping technical followers to perform effectively in their tasks, and in systems engineering this means facilitating pathways of communication between different areas, encouraging attitudes and behaviours that promote integrated perspectives.

Authentic Leadership

Somewhat in contrast to the principle of leading by adapting style, and thus in effect "acting the part", research on leaders being 'authentic' evaluates the effectiveness of staying true to one's own natural style. Successful authentic leaders are described as positive, leading from the heart, concerned with ethics, building on trust, motivating people to achieve challenging tasks. According to the authentic leadership literature (e.g., Gardner et al., 2011; Walumbwa et al., 2008), authentic leaders display four types of behaviours. These include balanced processing (taking evidence from all sides), internalized moral perspective (driven more by morality than external pressures, relational transparency (openly sharing thoughts and feelings), and self awareness (understanding of self and how others view them) (Gardner et al., 2011). These behaviours are likely to lead to a team having trust in the leader, which will be important in a technical context where achieving the right outcome, safely, in a complex situation is paramount.

Allied to authentic leadership in terms of behaviours, is the concept of Servant Leadership, described as having seven key practices: self-awareness, listening, inverting the pyramid (leadership hierarchy), developing your colleagues, coaching not controlling, unleashing the energy and intelligence of others, and foresight. Keith (2012) and Sipe and Frick (2009) have a similar list: servant-leaders are individuals of character, put people first, are skilled communicators, are compassionate collaborators, use foresight, are systems thinkers, and exercise moral authority.

The servant leadership elements of Empowerment, Standing Back / Sharing Credit, Courage / Risk Taking, Humility, Authenticity, and Stewardship were shown to have a statistically significant correlation with innovation output from engineering teams when applied at a frontline team leadership level (McCleave and Capella, 2015).

Complexity Leadership and the Leadership Process

Authentic and servant leadership styles place a leader in the role of a facilitator, rather than a director; someone who can leverage the capabilities of the team and create synergistic benefits. This perspective is taken a step further in the model of leadership that comes from complexity theory.

Complexity Leadership describes leadership as promoting emergent adaptive outcomes from organisations (such as learning and innovation). Organisations are considered to be complex adaptive systems and leadership can take three forms: administrative, adaptive and enabling. Each form will vary itself according to its locus in an organisational hierarchy. The complex adaptive functions provide the adaptive capability while the bureaucratic functions provide the coordinating structures. Leadership should disentangle these two types of functions in a way that enhances the effectiveness of the organisation (Marion & Uhl-Bien, 2001). In this model, leadership is mostly about developing

interactions.

Complexity leadership is differentiated from leaders as individuals, because in some cases leadership is about a function rather than a person. In a technical situation such as a Systems Engineering team, this will be an important consideration, as different people will have technical expertise and will be required to provide leadership in areas such as understanding, challenging and communicating. Systems engineering teams consist of members from diverse disciplines with diverse interests. Silos of self-interest must be broken down (or at least effective communication among silos most be established and a balance between global system concerns and provincial disciplinary interests must be maintained.)

Manz and Sims (1989) also see leadership as a process, but they focus on self-leadership within each individual more than the behaviours and actions of a few select people designated as formal leaders in an organization. With this perspective, most people have some contribution to leadership.

Followership

Equally important is the concept of followership. A leader can only lead with effective followers. In technical situations, where a distributed process of leadership may be needed, this is especially important. The study of followership is much less developed than that of leadership, although they are two sides of the same coin. Uhl-Bien et al (2014) have conducted a review of the literature to date and identify two theoretical frameworks for understanding followership: a role-based approach and a process approach. They warn against too much focus on a leader role and not enough on the leadership process, and suggest that understanding followership can help with:

- Recognizing the importance of follower roles, following behaviours, and the leadership process.
- Understanding leadership processes and its outcomes as a function of leaders and followers
- Identifying effective followership behaviours.
- Embedding context in the leadership process.
- Recognizing that leadership can flow in all directions
- Understanding why and how managers are not always able to co-construct leadership with their subordinates
- Followership development

This perspective is supportive of a distributed leadership function, and is helpful for supporting people who have leadership roles as a consequence of their technical know-how rather than their desire to lead or comfort with doing so.

Associated with followership development is the nature of motivation within the individuals that the leader wishes to influence. The term ‘motivation’ has been used to describe a range of possible causes of behaviour, and no single theory can explain all situations. A useful distinction, however, is the difference between ‘intrinsic’ and ‘extrinsic’ motivation. The former relates to factors arising from emotions, ambitions, expectations and other internal states of an individual, and tends to be the focus of transformational leaders (see section 1.3). The latter relates to factors arising from external factors such as threats, rewards, and social pressure, and tends to be the focus of transactional leaders (also in section 1.3). It is important to recognise that there are cultural and professional differences in how powerful internal and external causes of motivation are. One famous model of motivation by Maslow (1943), the ‘Hierarchy of Needs’, is useful to assess a range of potential factors, but does not have scientific validity and is based on a rather narrow Western 20th Century perspective. For example, it does not explain why people are willing to undergo physical hardship to conquer higher level challenges; or why some cultures are collectivist while others are individualistic. (A useful review of these culture differences can be found in: Triandis et al., 1988).

A more actionable approach to motivation emphasizes an individual’s mental model of what is important (valence), what their own role is in achieving it (instrumentality), and how able they are to achieve it (expectancy). This was first described by Vroom (1964) and has led to the concept of ‘empowering’ individuals (e.g. Conger and Kanungo, 1988). The Path-Goal model of leadership (section 1.6) aims to facilitate performance by addressing these aspects of motivation. This approach to motivation, called Expectancy Theory, can help leaders understand how to motivate

employees through challenge and self-belief (Isaac, Zerbe, and Pitt, (2001).

An attempt to understand motivation at the organisational level has led to the concept of 'organisational energy' (Cole, Bruch and Vogel, 2005). According to the existing overall energy type in an organisation, a leader should adopt a different motivational strategy to achieve the optimum 'productive' energy, which is described as high intensity and positive. A resignative energy (low intensity, negative) requires the development of a vision, empowerment and challenge. A corrosive energy (high intensity, negative) requires better communication and the development of trust. A comfortable energy (low intensity, positive) requires the identification of an external threat

Competencies

Leadership competencies are the knowledge and skills required by individuals and teams for making leadership effective. Sometimes traits and other individual differences are added to skills and knowledge to create a 'Competency Framework' for the leadership characteristics needed for a role. Communication, managing staff by supporting and providing feedback, and emotional competence are often featured in these frameworks. It is important to distinguish between those characteristics that are learned and those that are based on traits. Learned competencies can be enhanced through personal development; innate individual differences could be acquired for a role through personnel selection (although selection based on personality is not recommended: see section 1.2). As indicated above, leadership depends on many behaviours, including matching style to situations, effective followership, and individual leadership.

A number of roles will be required in a team, and ideally these may be distributed to individuals with the apposite competencies. Emotional competence has been the focus of much recent research and some studies show a strong correlation with effective leadership (e.g. Cavallo and Brienza, 2006, who used the Emotional Competence Inventory©).

Daniel Goleman has extended and publicised the concept of emotional intelligence (an innate characteristic) and the competencies (skills that can be learned) that put it into practice. He describes how emotional aptitudes can preserve relationships, protect our health and improve our success at work (Goleman, 1998).

Goleman differentiates 5 main categories of competence. The first three are about self-management and the last two are about being effective in relationships. 1. Self-awareness: accurate self-assessment, emotional awareness and self-confidence 2. Self-regulation: innovation, adaptability, conscientiousness, trustworthiness and self-control 3. Motivation: optimism, commitment, initiative and achievement, drive 4. Empathy: developing others, service orientation, political awareness, diversity, active listening and understanding others 5. Social skills: communication, influence, conflict management, leadership, bond building, collaboration, cooperation and team capabilities Emotional Intelligence is most associated with transformational and situational leadership.

Communication skills are also highlighted in most leader competency frameworks. These skills are about both communicating to other people and also listening and being communicated to by other people. Some skills are about engagement, others about sharing understanding. In particular, avoiding hidden assumptions and understanding others' perspectives are important. Communication can take place in many ways, especially with the help of IT and social media. Each mode of communication has advantages and disadvantages. Consideration should be given as to how important it is to have a face to face communication (usually better, but especially for complex matters and when emotions are involved). Although this takes more time and effort, it will often save time and effort in the longer term by reducing misunderstandings and negative emotions. Nikoi (2014) presents a collection of studies that investigated the way in which communication works across media and teams.

Communication may be synchronous or asynchronous, broadcast or individual, dialogue or one-way. Bowman (2004) has a useful summary of the advantages and disadvantages of different communication channels.

Some competencies that are often associated in the literature with good leadership are listed in Table1. The relevance of these will depend on the style and the situation/context.

Some commonly cited attributes of effective leaders are listed in Table 1 below.

Table 1. Attributes of Effective Leaders (Fairley 2009).
Reprinted with permission of the IEEE Computer Society. All other rights are reserved by the copyright owner.

Listening carefully	Maintaining enthusiasm
Delegating authority	Saying “thank you”
Facilitating teamwork	Praising team for achievements
Coordinating work activities	Accepting responsibility for shortcomings
Facilitating communication	Coaching and training
Making timely decisions	Indoctrinating newly assigned personnel
Involving appropriate stakeholders	Reconciling differences and resolving conflicts
Speaking with individual team members on a frequent basis	Helping team members develop career paths and achieve professional goals
Working effectively with the project/program manager and external stakeholders	Reassigning, transferring, and terminating personnel as necessary

Characteristics that result in effective leadership of systems engineering activities include behavioral attributes, leadership style, and communication style. In addition, a team leader for a systems engineering project or program has management responsibilities that include, but are not limited to: developing and maintaining the systems engineering plan, as well as establishing and overseeing the relationships between the project/program manager and project/program management personnel.

Implications for technical leadership in systems engineering

Leadership can have a significant impact on engineering performance (Kolb, 1995) and resilience (Flin, 2006). The models and styles of leadership described above emphasize the power of social skills: the ability to relate to and connect with other people. This appears to be particularly true for the sorts of situations that system engineering leaders are likely to find themselves in: working on complex problems with other professionals who are willing to follow but need to be confident in the leader’s technical skill and trustworthiness. The technical leader should possess not only essential technical knowledge but also should have positive values, high levels of ethics, morality, leading from the heart, personal capabilities, out-of-the-box thinking, interpersonal skills, etc. (Lloyd-Walker and Walker, 2011).

In a systems engineering context it is useful to recognise that different leadership functions may be distributed across a team. Some leadership functions will be knowledge focussed, but it may be necessary to have a ‘facilitator’ (complexity) leader to ensure that the team follows the most appropriate leadership at any time. Each organisation will have particular leadership requirements, which should be articulated in a behavioural framework in order to identify the most effective leadership styles and competencies, and where and how they should be applied.

Leadership capability for systems engineers should therefore be seen as a distributed capability to be developed across engineers. NASA takes a systems approach to developing leadership in their Systems Engineering Leadership Development Program (SELDP). They define technical leadership as the ‘art’ of systems engineering. Technical leadership includes broad technical domain knowledge, engineering instinct, problem solving, creativity, and the leadership and communication skills needed to develop new missions and systems. It focuses on systems design and technical integrity throughout the life cycle. A system’s complexity and the severity of its constraints drive the need for systems engineering leadership (Williams and Reyes, 2012).

Selecting leaders by promoting the best technical performers or the most ambitious candidates is not an effective way of ensuring good leadership in an organisation or programme. For this reason, companies such as General Electric, Motorola, Toyota, Unilever, Raytheon, and Northrop Grumman use internal leadership academies to develop their leadership capability according to their needs (Daniels, 2009). A role model approach may be effective only if the appropriate role model is paired with a candidate, with good leadership characteristics that are valid for the situation (Yukl, 2012).

More effective approaches would involve developing competencies that can be learned through example, experience and reflection. The most effective methods will depend on the competencies needed, the type of organisation, and the opportunities. They could include coaching, mentoring, shadowing, 'assistant-to' trial periods, and career management to provide experience (e.g. Fast-track).

There must also be an element of self-development: systems engineers should recognise the impact that people (or 'soft') issues have on the performance of a technical team and organisation and learn how to adjust their own behaviour and facilitate the behaviour of others.

Behavioral Attributes

Behavioral attributes are habitual patterns of behavior, thought, and emotion that remain stable over time (Yukl 2013). Positive behavioral attributes enable a systems engineering leader to communicate effectively and to make sound decisions, while also taking into consideration the concerns of all stakeholders. Desirable behavioral attributes for a systems engineering leader include characteristics such as (Fairley 2009):

- Aptitude - This is exhibited by the ability to effectively lead a team. Leadership aptitude is not the same as knowledge or skill but rather is indicative of the ability (either intuitive or learned) to influence others. Leadership aptitude is sometimes referred to as charisma or as an engaging style.
- Initiative - This is exhibited by enthusiastically starting and following through on every leadership activity.
- Enthusiasm - This is exhibited by expressing and communicating a positive, yet realistic attitude concerning the project, product, and stakeholders.
- Communication Skills - These are exhibited by expressing concepts, thoughts, and ideas in a clear and concise manner, in oral and written forms, while interacting with colleagues, team members, managers, project stakeholders, and others.
- Team Participation - This is exhibited by working enthusiastically with team members and others when collaborating on shared work activities.
- Negotiation - This is the ability to reconcile differing points of view and achieve consensus decisions that are satisfactory to the involved stakeholders.
- Goal Orientation – This involves setting challenging but not impossible goals for oneself, team members, and teams.
- Trustworthiness - This is demonstrated over time by exhibiting ethical behavior, honesty, integrity, and dependability in taking actions and making decisions that affect others.

Weakness, on the other hand, is one example of a behavioral attribute that may limit the effectiveness of a systems engineering team leader.

Personality Traits

“Personality traits” was initially introduced in the early 1900's by Carl Jung, who published a theory of personality based on three continuums: introversion-extroversion, sensing-intuiting, and thinking-feeling. According to Jung, each individual has a dominant style which includes an element from each of the three continuums. Jung also emphasized that individuals vary their personality traits in the context of different situations; however, an individual’s dominant style is the preferred one, as it is the least stressful for the individual to express and it is also the style that an individual will resort to when under stress (Jung 1971). The Myers-Briggs Type Indicator (MBTI),

developed by Katherine Briggs and her daughter Isabel Myers, includes Jung's three continuums, plus a forth continuum of judging-perceiving. These four dimensions characterize 16 personality styles for individuals designated by letters, such as ISTP (Introverted, Sensing, Thinking, and Perceiving). An individual's personality type indicator is determined through the answers the person has provided on a questionnaire (Myers 1995) combined with the individuals self-assessment which is done one to one with a qualified practitioner or in a group setting. MBTI profiles are widely used by coaches and counselors to help individuals assess how their personality type will affect how they might react in a particular profession and make suggestions about which professions might suit their individual preferences. It should never be used to decide which profession would be "most comfortable and effective" as the MBTI measures preference not ability. The MBTI has also been applied to group dynamics and leadership styles. Most studies indicate that groups perform better when a mixture of personality styles work together to provide different perspectives. Some researchers claim that there is evidence that suggests that leadership styles are most closely related to an individual's position on the judging-perceiving scale of the MBTI profile (Hammer 2001). Those on the judging side of the scale are more likely to be "by the book" managers, while those on the perceiving side of the scale are most likely to be "people-oriented" leaders. "Judging" in the MBTI model does not mean judgmental; rather, a judging preference indicates a quantitative orientation and a perceiving preference indicates a qualitative orientation. The MBTI has its detractors (Nowack 1996); however, MBTI personality styles can provide insight into effective and ineffective modes of interaction and communication among team members and team leaders. For example, an individual with a strongly Introverted, Thinking, Sensing, and Judging personality index (ITSJ) may have difficulty interacting with an individual who has a strongly Extroverted, Intuiting, Feeling, Perceiving personality index (ENFP).

Leadership Styles and Communication Styles

There is a vast amount of literature pertaining to leadership styles and there are many models of leadership. Most of these leadership models are based on some variant of Jung's psychological types. One of the models, the Wilson Social Styles, integrates leadership styles and communication styles (Wilson 2004). The Wilson model characterizes four kinds of leadership styles:

- Driver leadership style - This is exhibited when a leader focuses on the work to be accomplished and on

specifying how others must do their jobs.

- Analytical-style leadership - This emphasizes collecting, analyzing, and sharing data and information. An

analytical leader asks others for their opinions and recommendations to gather information.

- Amiable leadership style – This is characterized by emphasis on personal interactions and on asking others

for their opinions and recommendations.

- Expressive leadership style – Like the amiable style, this also focuses on personal relationships, but an

expressive leader tells others rather than asking for opinions and recommendations. When taken to extremes, each of these styles can result in weakness of leadership. By focusing too intently on the work, "drivers" can provide too much or too little guidance and direction. Too little guidance occurs when the individual is preoccupied with her or his personal work, while too much guidance results in micromanagement, which limits the personal discretion for team members. Drivers may also be insensitive to interpersonal relationships with team members and others. Analytical leaders may provide too much information or may fail to provide information that is obvious to them, but not their team members. They do not like to discuss things they already know or that are irrelevant to the task at hand. Like driver-style leaders, they may be insensitive to interpersonal relationships with other individuals. Amiable leaders focus on interpersonal relationships in order to get the job done. They may exhibit a dislike of those who fail to interact with them on a personal level and may show little concern for those who show little personal interest in them. Expressive leaders also focus on interpersonal relationships. In the extreme, an expressive leader may be more interested in stating their opinions than in listening to others. Additionally, they may play favorites and ignore those

who are not favorites. While these characterizations are gross oversimplifications, they serve to illustrate leadership styles that may be exhibited by systems engineering team leaders. Effective team leaders are able to vary their leadership style to accommodate the particular context and the needs of their constituencies without going to extremes; but as emphasized by Jung, each individual has a preferred comfort zone that is least stressful and to which an individual will resort during times of added pressure.

Communication Styles

An additional characterization of the Wilson model is the preferred style of communication for different leadership styles, which is illustrated by the dimensions of assertiveness and responsiveness.

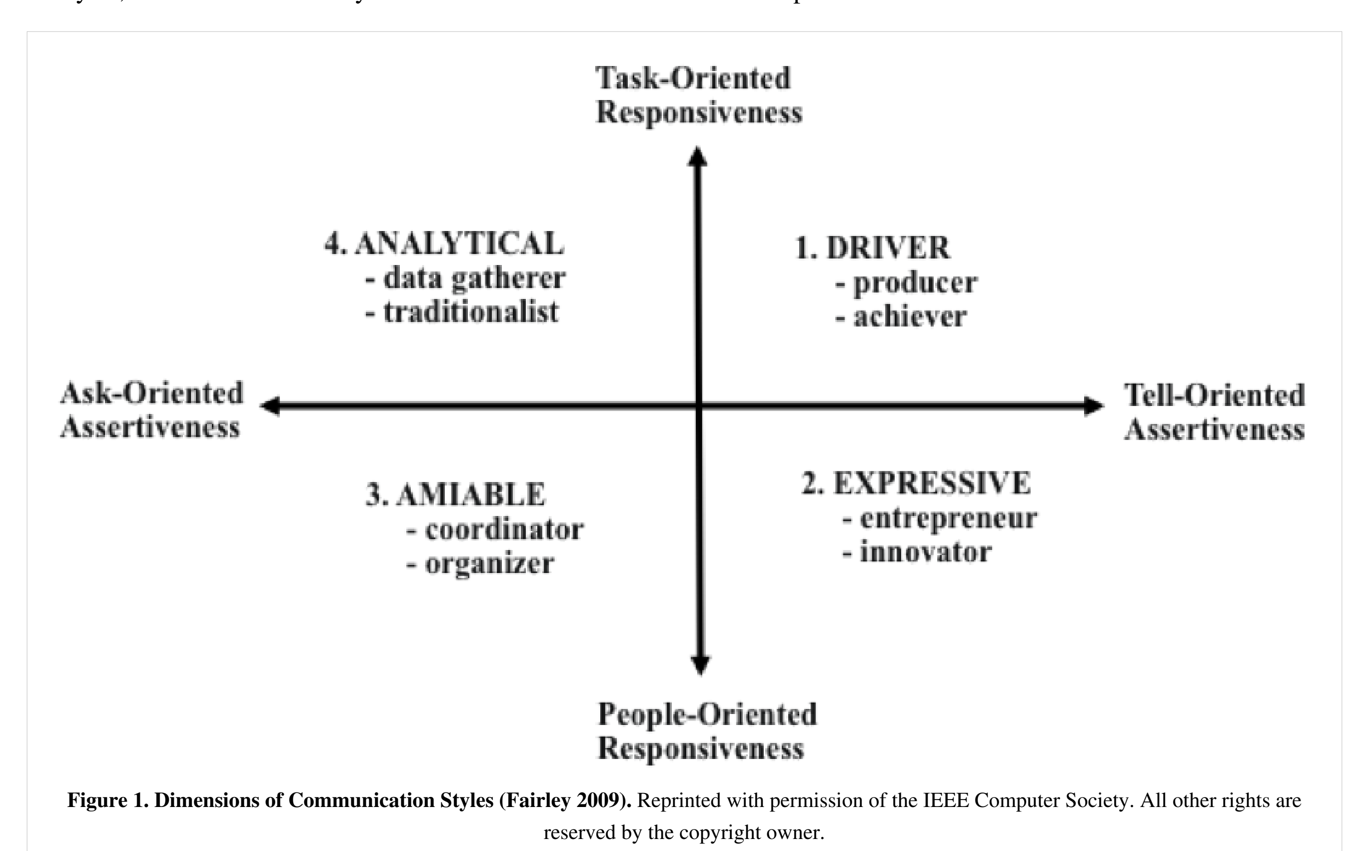


Figure 1. Dimensions of Communication Styles (Fairley 2009). Reprinted with permission of the IEEE Computer Society. All other rights are reserved by the copyright owner.

Task-oriented assertiveness is exhibited in a communication style that emphasizes the work to be done rather than on the people who will do the work, while the people-oriented communication style addresses personnel issues first and tasks secondly. A tell-oriented communication style involves telling rather than asking, while an ask-oriented assertiveness emphasizes asking over telling. Movies, plays, and novels often include caricatures of extremes in the assertiveness and responsiveness dimensions of Wilson communication styles. An individual’s communication style may fall anywhere within the continuums of assertiveness and responsiveness, from extremes to more moderate styles and may vary considering the situation. Examples include:

- Driver communication style exhibits task-oriented responsiveness and tell-oriented assertiveness.
- Expressive communication style shares tell-oriented assertiveness with the driver style, but favors people-oriented responsiveness.
- Amiable communication style involves asking rather than telling (as does the analytical style) and emphasizes people relationships over task orientation (as does the expressive style).
- Analytical communication style exhibits task-oriented responsiveness and ask-oriented assertiveness.

The most comfortable communication occurs when individuals share the same communication styles or share adjacent quadrants in Figure 1. Difficult communication may occur when individuals are in diagonal quadrants; for example, communication between an extreme amiable style and an extreme driver style. Technical leaders and others

can improve communications by being aware of different communication styles (both their own and others) and by modifying their communication style to accommodate the communication styles of others.

Management Responsibilities

Leading a systems engineering team involves communicating, coordinating, providing guidance, and maintaining progress and morale. Managing a project, according to the PMBOK® Guide (PMBOK 2013), involves application of the five process groups of project management: initiating, planning, executing, monitoring and controlling, and closing. Colloquially, systems engineering project/program management is concerned with making and updating plans and estimates, providing resources, collecting and analyzing product and process data, working with the technical leader to control work processes and work products, as well as managing the overall schedule and budget. Good engineering managers are not necessarily good technical leaders and good technical leaders are not necessarily good engineering managers; the expression of different personality traits and skill sets is required. Those who are effective as both managers and leaders have both analytical and interpersonal skills, although their comfort zone may be in one of managing or leading. Two management issues that are typically the responsibility of a systems engineering team leader are:

- Establishing and maintaining the division of responsibility among him or herself, the systems engineering team leader, and the project/program manager.
- Developing, implementing, and maintaining the systems engineering plan (SEP).

Relationships between systems engineering and project management are addressed in the Part 6 Knowledge Area (KA) of the SEBoK, Systems Engineering and Project Management. Also, see the Part 5 Knowledge Area Enabling Teams for a discussion of the relationships between a project/program manager and a systems engineering technical leader.

The System Engineering Plan (SEP) is, or should be, the highest-level plan for managing the Systems Engineering effort and the technical aspects of a project or program. It defines how a project will be organized and conducted in terms of both performing and controlling the Systems Engineering activities needed to address a project's system requirements and technical content. It can have a number of secondary technical plans that provide details on specific technical areas and supporting processes, procedures, tools. Also, see the Planning article in Part 3, which includes a section on Systems Engineering Planning Process Overview.

In United States DoD acquisition programs, the System Engineering Plan (SEP) is a Government produced document which assists in the development, communication, and management of the overall systems engineering (SE) approach that guides all technical activities of the program. It provides direction to developers for program execution. The developer uses the SEP as guidance for producing the System Engineering Management Plan (SEMP), which is a separate document and usually a contract deliverable that aligns with the SEP. As the SEP is a Government produced and maintained document and the SEMP is a developer/contractor developed and maintained document, the SEMP is typically a standalone, coordinated document.

The following SEP outline from (ODASD 2011) serves as an example.

1. **Introduction – Purpose and Update Plan**
2. **Program Technical Requirements**
    1. Architectures and Interface Control
    2. Technical Certifications
3. **Engineering Resources and Management**
    1. Technical Schedule and Schedule Risk Assessment
    2. Engineering Resources and Cost/Schedule Reporting
    3. Engineering and Integration Risk Management
    4. Technical Organization

   5. Relationships with External Technical Organizations
   6. Technical Performance Measures and Metrics
4. **Technical Activities and Products**
   1. Results of Previous Phase SE Activities
   2. Planned SE Activities for the Next Phase
   3. Requirements Development and Change Process
   4. Technical Reviews
   5. Configuration and Change Management Process
   6. Design Considerations
   7. Engineering Tools
5. **Annex A – Acronyms**

SEP templates are often tailored to meet the needs of individual projects or programs by adding needed elements and modifying or deleting other elements. A systems engineering team leader typically works other team members, the project/program manager (or management team), and other stakeholders to develop the SEP and maintain currency of the plan as a project evolves. Some organizations provide one or more SEP templates and offer guidance for developing and maintaining an SEP. Some organizations have a functional group that can provide assistance in developing the SEP.

References

Works Cited

Bennis, W.G. and Nanus, B. (1985) Leaders: Strategies for Taking Charge. Harper & Row

Bowman (2004) Business Communication: Managing Information and Relationships https://homepages.wmich.edu/~bowman/channels.html

Burns, J.M. (1978) Leadership. New York: Harper & Row

Cavallo, K. and Brienza, D. (2006) Emotional Competence and Leadership Excellence at Johnson & Johnson, The Emotional Intelligence and Leadership Study, Europe's Journal of Psychology, Vol.2, No 1.

Cole, M. S., Bruch, H., & Vogel, B. (2005). Development and validation of a measure of organizational energy. Academy of Management Best Paper Proceedings. In Weaver, K. M. (Ed.), Proceedings of the Sixty-fourth Annual Meeting of the Academy of Management (CD), ISSN 1543-8643.

Conger, J.A. and Kanungo, R.N. (1988) The Empowerment Process: Integrating Theory and Practice, The Academy of Management Review, Vol. 13, No. 3, pp. 471-482

Daniels, C. B. (2009) Improving Leadership in a Technical Environment: A Case Example of the ConITS Leadership Institute, Engineering Management Journal, 21, pp. 47-52

Drath, W. H., and Palus, C. J. (1994). Making common sense: Leadership as meaning-making in a community of practice. Greensboro, NC: Center for Creative Leadership.

Fairley, R.E. (2009) Managing and Leading Software Projects. Hoboken, NJ, USA: John Wiley & Sons

Fiedler, F. E. (1964) A Contingency Model of Leadership Effectiveness, Advances in Experimental Social Psychology, 1, 149–190

Flin, R. (2006) Erosion of Managerial Resilience: From Vasa to NASA. In Hollnagel, E. Woods, D.D., and Levenson, N. (Eds), Resilience Engineering Concepts and Precepts, pp. 223-233. Aldershot: Ashgate

Franz, C. R. (1985) User Leadership in the Systems Development Life Cycle: A Contingency Model. Journal of Management Information Systems, 2 (2), 5-25

Frick, D. (2009) Seven Pillars of Servant Leadership: Practicing the Wisdom of Leading by Serving, New Jersey: Paulist Press)

Gardner, W.L., Cogliser, C.C., Davis, K.M., & Dickens, M.P. (2011). Authentic leadership: A review of the literature and research agenda. Leadership Quarterly, 22, 1120-1145

Goleman D. (1998) The emotionally competent leader. Health Forum Journal, 41(2), 38- 76

Gurven, M., Von Rueden, C., and Kaplan, H., Vie, M.L., (2013) How Universal Is the Big Five? Testing the Five-Factor Model of Personality Variation Among Forager–Farmers in the Bolivian Amazon., Journal of Personality and Social Psychology, Vol. 104, No. 2, 354–370

Hersey, P., Blanchard, K.H. and Johnson, D.E. (2001) Management of Organisational Behaviour Leading Human Resources. New Jersey: Prentice Hall

Herzberg, F., Mausnek, B., And Snyderman, B. (1959) The Motivation to Work (Second Edition). New York: John Wiley and Sons, 1959.

House, R. J., and Mitchell, R. R. (1974) Path-goal theory of leadership.

Journal of Contemporary Business, 3(4), pp. 81-98

Isaac, R. G., Zerbe, W.J., & Pitt, D. C. (2001) Leadership and motivation: The effective application of expectancy theory. Journal of Managerial Issues, 13(2), 212-226

Judge, T. A., Bono, J. Y., Ilies, R., & Gerhardt, M. W. (2002). Personality

and leadership: A qualitative and quantitative review. Journal of Applied

Psychology, 87, 765–780

Keith, K. (2012) The Case for Servant-Leadership, Honolulu, Hawaii: Terrace Press, Second Edition

Kolb, J. A. (1995). Leader behaviours affecting team performance: Similarities and differences between leader/member assessments. Journal of Business Communication, 32, 233-248

Lewin, K., LIippit, R. and White, R.K. (1939). Patterns of aggressive behavior in experimentally created social climates. Journal of Social Psychology, 10, 271-301

Lloyd-Walker, B. and Walker, D. (2011) Authentic Leadership for 21st Century Project Delivery, International Journal of Project Management, 29, pp 383-395

Sipe, J. and Frick, D. (2009) The Seven Pillars of Servant Leadership, Practicing the wisdom of leading by serving. Paulist Press

Triandis, H.C., Bontempo, R., Villareal, M.J. , Asai, M. and Lucca, N (1988) Individualism and Collectivism: Cross-Cultural Perspectives on Self-Ingroup Relationships, Journal of Personality and Social Psychology, Vol.5 4, No, 2. 323-338

Manz, C. C., and Sims, H. P., Jr. (1989). SuperLeadership. NY: Prentice Hall Press

McCleave, E. B., and Capella U. (2015) A correlational analysis of frontline leaders as drivers of technical innovation in the aerospace industry based on the servant leadership theory.

US Dissertation Abstracts International Section A: Humanities and Social Sciences, Vol 75(8-A)(E)

Marion, R., and Uhl-Bien, M. (2001) Leadership in complex organizations, The Leadership Quarterly, 12, pp. 389–418

Maslow, A.H. (1943) A theory of human motivation, Psychological Review 50 (4) 370–96

National Research Council (1991) In The Mind's Eye, Washington, DC: National Academy of Science

Nikoi, E. (Ed) (2014) Collaborative Communication Processes and Decision Making in Organizations. Hershey, PA: Business Science Reference

Stogdill, R.M. (1948) Personal Factors Associated with Leadership: a Survey of the Literature. Journal of Psychology, Vol. 25.

Toor, S. R. and Ofori, G., (2008) Leadership vs. Management: How they are different, and why!

Journal of Leadership and Management in Engineering, 8(2), 61- 71

Uhl-Bien, M., Riggio, R.E. , Lowec, K.B. , Melissa K. Carstend, M.K. (2014) Followership theory: A review and research agenda, Leadership Quarterly 25th Anniversary Issue, The Leadership Quarterly, Volume 25, Issue 1, Pages 83–104

Valle, S., & Avella, L. (2003) Cross-functionality and leadership of the new product development teams. European Journal of Innovation Management, 6(1), 32 – 47

Vroom V H. (1964) Work and motivation. New York: Wiley

Walumba, F.O., Avolio, B.J., Gardner, W.L., Wernsing, T.S., and Peterson, S.J., (2008) Authentic Leadership: Development and Validation of a Theory-Based Measure, Journal of Management 34:1, pp. 89-126

Williams, C.R. and Reyes , A. (2012) Developing Systems Engineers at NASA Global Journal of Flexible Systems Management, 13(3), 159–164

Yukl, G.A. (2012). Leadership in Organizations. 8th Ed. Upper Saddle River, NJ: Prentice Hall

Primary References

Fairley, R.E. 2009. Managing and Leading Software Projects. Hoboken, NJ, USA: John Wiley & Sons.

Myers, I.B., and P.B. Myers. 1995. Gifts Differing: Understanding Personality Type, 2nd ed. Mountain View, CA: Davies-Black Publishing under special license from CPP, Inc.

Wilson, Larry. 2004. The Social Styles Handbook. Belgium: Nova Vista Publishing.

Barrett, D.J. (2006), Leadership communication. McGraw Hill Education, Boston

Bass, B. M., & Bass, R. (2008). The Bass Handbook of Leadership: Theory, Research, and Managerial Applications. New York: Free Press

Bennis, W. (2003) On Becoming a Leader. Perseus Publishing

Northouse, P. G. (2007). Leadership theory and practice (4th ed.). Thousand Oaks, CA: Sage.

Additional References

Bass, B. M., & Avolio, B. J. (1994), Improving organizational effectiveness through transformational leadership, Thousand Oaks, CA: Sage

Fiedler, F. E. (1964) A contingency model of leadership effectiveness. In L. Berkowitz (Ed.), Advances in experimental social psychology (Vol. 1). New York: Academic Press

Lowe, K. B., Kroeck, K. G., & Sivasubramaniam, N. (1996). Effectiveness correlates of transformational and transactional leadership: A meta-analytic review of the MLQ literature. The Leadership Quarterly, 7(3), 385-415

Pandya. K. D. (2014) Key Competencies of Project Leader Beyond the Essential Technical Capabilities, IUP Journal of Knowledge Management, Vol. 12 Issue 4, 39-48.

Ram, C., Drotter, S. and Noel, J. (2001) The Leadership Pipeline: How to Build the Leadership Powered Company. Wiley: Jossey-Bass

---

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Enabling Individuals

Enabling Individuals

This knowledge area focuses on enabling an individual to perform SE, and addresses the roles of individuals in the SE profession, how individuals are developed for and assessed in these roles, and what ethical behavior is expected of them. Once an individual is enabled to perform SE using the techniques described here, the individual can apply the knowledge found in Part 3, Systems Engineering and Management, about how to perform SE.

Part 5, Enabling Systems Engineering, to which this knowledge area belongs, explores how systems engineering (SE) is enabled at three levels of organization: the business or enterprise, the team, and the individual. Ultimately, individuals perform SE tasks within a team or business.

For the sake of brevity, the term “business” is used to mean “business or enterprise” throughout most of this knowledge area. For a nuanced explanation of what distinguishes a business from a enterprise, see Enabling Systems Engineering.

Topics

Each part of the SEBoK is composed of knowledge areas (KAs]. Each KA groups topics together around a theme related to the overall subject of the part. This KA contains four topics:

- Roles and Competencies discusses allocation of SE roles, which sets of competencies (glossary) correspond to particular roles, and what competency models are current in the SE world.
- Assessing Individuals discusses how to determine the level of individual proficiency and quality of performance.
- Developing Individuals explains how SE competency is acquired.
- Ethical Behavior describes the ethical standards that apply to individuals and organizations.

Context

The following brief review of terms and concepts provides context for the topics in this knowledge area.

Individuals, Teams, Businesses, and Enterprises

The ability to perform SE resides in individuals, teams, and businesses. An expert systems engineer possesses many competencies at a high level of proficiency, but no one can be highly proficient in all possible competencies. Collectively, a team and a business might possess all needed competencies at a high level of proficiency. A business performs the full range of SE roles, may have dedicated functions to perform specific SE roles, and may have a strategy for combining individual, team, and business abilities to execute SE on a complex activity. Individuals within the business may be responsible for performing one or more roles.

For descriptions of SE roles and competencies from the literature, see Roles and Competencies.

Competency, Capability, Capacity, and Performance

The final execution and performance of SE is a function of competency, capability, and capacity. There is some complexity here. For example:

- There is disagreement in the literature about whether the term competency applies to the individual level only, or can be correctly used at the team, project, and enterprise levels as well.
- Capability encompasses not just human capital, but processes, machines, tools, and equipment as well. Even if an individual has an outstanding level of competency, having to perform within a limited timeframe might degrade the results. Capacity accounts for this.

Systems Engineering Competency

Competency is built from knowledge, skills, abilities, and attitudes (KSAA). What is inherent in an individual may be subsequently developed through education, training, and experience. Traditionally, SE competencies have been developed primarily through experience, but recently, education and training have taken on a much greater role.

SE competency must be viewed through its relationships to the systems life cycle, the SE discipline, and the domain in which the engineer practices SE.

Competency Models

SE competency models can be used to explicitly state and actively manage the SE competencies within in an organization.

Competency models for SE typically include

- technical KSAAs;
- “soft” KSAAs such as leadership and communications;
- KSAAs that focus on the domains within which SE is to be practiced;
- a set of applicable competencies; and
- a scale for assessing the level of individual proficiency in each competency (often subjective, since proficiency is not easily measured).

See Roles and Competencies for descriptions of publicly available SE competency models.

References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Roles and Competencies

Enabling individuals to perform systems engineering (glossary) (SE) requires an understanding of SE competencies, roles, and tasks; plus knowledge, skills, abilities, and attitudes (KSAA). Within a business (glossary) or enterprise, SE responsibilities are allocated to individuals through the definition of SE roles associated with a set of task. For an individual, a set of KSAAs enables the fulfillment of the competencies needed to perform the tasks associated with the assigned SE role. SE competencies reflect the individual's KSAAs, which are developed through education, training, and on-the-job experience. Traditionally, SE competencies build on innate personal qualities and have been developed primarily through experience. Recently, education and training have taken on a greater role in the development of SE competencies.

Relationship of SE Competencies and KSAAs

There are many ways to define competency. It can be thought of as a measure of the ability to use the appropriate KSAAs to successfully complete specific job-related tasks (Whitcomb, Khan, White 2014). Competencies align with the tasks that are expected to be accomplished for the job position (Holt and Perry 2011). KSAAs belong to the individual. In the process of filling a position, organizations have a specific set of competencies associated with tasks that are directly related to the job. A person possesses the KSAAs that enable them to perform the desired tasks at an acceptable level of competency.

The KSAAs are obtained and developed from a combination of several sources of learning including education, training, and on-the-job experience. By defining the KSAAs in terms of a standard taxonomy, they can be used as learning objectives for competency development (Whitcomb, Khan, White 2014). Bloom's Taxonomy for the cognitive and affective domains provides this structure (Bloom 1956, Krathwohl 2002). The cognitive domain includes knowledge, critical thinking, and the development of intellectual skills, while the affective domain describes growth in awareness, attitude, emotion, changes in interest, judgment, and the development of appreciation (Bloom 1956). The affective does not refer to additional topics which a person learns about, but rather to a transformation of the person in relation to the original set of topics learned. Cognitive and affective processes within Bloom's taxonomic classification schema refer to levels of observable actions, which indicate learning is occurring. Bloom's Taxonomy for the cognitive and affective domains define terms as categories of levels that can be used for consistently defining KSAA statements (Krathwohl 2002):

Cognitive Domain

- Remember
- Understand
- Apply
- Analyze
- Evaluate
- Create

Affective Domain

- Receive
- Respond
- Value
- Organize
- Characterize

Both cognitive and affective domains should be included in the development of systems engineering competency models, because the cognitive domain learning concerns the consciously developed knowledge about the various subjects and the ability to perform tasks, whilst the affective learning concerns the interest in or willingness to use

particular parts of the knowledge learned and the extent to which the systems engineer is characterized by taking approaches which are inherently systemic. Using the affective domain in the specification of KSAAs, is also important as every piece of information we process in our brains goes through our affective (emotional) processors before it is integrated by our cognitive processors (Whitcomb and Whitcomb 2013).

SE Competency Models

Contexts in which individual competency models are typically used include

- **Recruitment and Selection:** Competencies define categories for behavioral event interviewing (BEI), increasing the validity and reliability of selection and promotion decisions.
- **Human Resources Planning and Placements:** Competencies are used to identify individuals to fill specific positions and/or identify gaps in key competency areas.
- **Education, Training, and Development:** Explicit competency models let employees know which competencies are valued within their organization. Curriculum and interventions can be designed around desired competencies.

Commonality and Domain Expertise

No single individual is expected to be proficient in all the competencies found in any model. The organization, overall, must satisfy the required proficiency in sufficient quantity to support business needs. Organizational capability is not a direct summation of the competency of the individuals in the organization, since organizational dynamics play an important role that can either raise or lower overall proficiency and performance. The articles Enabling Teams and Enabling Businesses and Enterprises explore this further.

SE competency models generally agree that systems thinking, taking a holistic view of the system that includes the full life cycle, and specific knowledge of both technical and managerial SE methods are required to be a fully capable systems engineer. It is also generally accepted that an accomplished systems engineer will have expertise in at least one domain of practice. General models, while recognizing the need for domain knowledge, typically do not define the competencies or skills related to a specific domain. Most organizations tailor such models to include specific domain KSAAs and other peculiarities of their organization.

INCOSE Certification

Certification is a formal process whereby a community of knowledgeable, experienced, and skilled representatives of an organization, such as the International Council on Systems Engineering (INCOSE), provides formal recognition that a person has achieved competency in specific areas (demonstrated by education, experience, and knowledge). (INCOSE nd). The most popular credential in SE is offered by INCOSE, which requires an individual to pass a test to confirm knowledge of the field, requires experience in SE, and recommendations from those who have knowledge about the individual's capabilities and experience. Like all such credentials, the INCOSE certificate does not guarantee competence or suitability of an individual for a particular role, but is a positive indicator of an individual's ability to perform. Individual workforce needs often require additional KSAAs for any given systems engineer, but certification provides an acknowledged common baseline.

Domain- and Industry-specific Models

No community consensus exists on a specific competency model or small set of related competency models. Many SE competency models have been developed for specific contexts or for specific organizations, and these models are useful within these contexts.

Among the domain- and industry-specific models is the Aerospace Industry Competency Model (ETA 2010), developed by the Employment and Training Administration (ETA) in collaboration with the Aerospace Industries Association (AIA) and the National Defense Industrial Association (NDIA), and available online. This model is designed to evolve along with changing skill requirements in the aerospace industry. The ETA makes numerous competency models for other industries available online (ETA 2010). The NASA Competency Management System (CMS) Dictionary is predominately a dictionary of domain-specific expertise required by the US National Aeronautics and Space Administration (NASA) to accomplish their space exploration mission (NASA 2009).

Users of models should be aware of the development method and context for the competency model they plan to use, since the primary competencies for one organization might differ from those for another organization. These models often are tailored to the specific business characteristics, including the specific product and service domain in which the organization operates. Each model typically includes a set of applicable competencies along with a scale for assessing the level of proficiency.

SE Competency Models — Examples

Though many organizations have proprietary SE competency models, published SE competency models can be used for reference. Table 1 lists information about several published SE competency models, and links to these sources are shown below in the references section. Each model was developed for a unique purpose within a specific context and validated in a particular way. It is important to understand the unique environment surrounding each competency model to determine its applicability in any new setting.

Table 1. Summary of Competency Models. (SEBoK Original)

Competency Model	Date	Author	Purpose	Development Method	Competency Model
INCOSE UK WG	2010	INCOSE	Identify the competencies required to conduct good systems engineering	INCOSE Working Group	(INCOSE 2010), (INCOSE UK 2010)
ENG Competency Model	2013	DAU	Identify competencies required for the DoD acquisition engineering professional.	DoD and DAU internal development	(DAU 2013)
NASA APPEL Competency Model	2009	NASA	To improve project management and systems engineering at NASA	NASA internal development - UPDATE IN WORK	(NASA 2009)
MITRE Competency Model	2007	MITRE	To define new curricula systems engineering and to assess personnel and organizational capabilities	Focus groups as described in (Trudeau 2005)	(Trudeau 2005), (MITRE 2007)
CMMI for Development	2007	SEI	Process improvement maturity model for the development of products and services	SEI Internal Development	(SEI 2007), (SEI 2004)

Other models and lists of traits include: Hall (1962), Frank (2000; 2002; 2006), Kasser et al. (2009), Squires et al. (2011), and Armstrong et al. (2011). Ferris (2010) provides a summary and evaluation of the existing frameworks for personnel evaluation and for defining SE education. Squires et al. (2010) provide a competency-based approach that can be used by universities or companies to compare their current state of SE capability development against a government-industry defined set of needs. SE competencies can also be inferred from standards such as ISO-15288 (ISO/IEC/IEEE 15288 2015) and from sources such as the INCOSE *Systems Engineering Handbook* (INCOSE 2012), the INCOSE Systems Engineering Certification Program, and CMMI criteria (SEI 2007). Whitcomb, Khan, and White describe the development of a systems engineering competency model for the United States Department

of Defense based on a series of existing competency models (Whitcomb, Khan, and White 2013; 2014).

To provide specific examples for illustration, more details about three SE competency model examples follow. These include:

- The International Council on Systems Engineering (INCOSE) UK Advisory Board model (INCOSE 2010), (INCOSE UK 2009);
- The DAU ENG model (DAU 2013); and
- The NASA Academy of Program/Project & Engineering Leadership (APPEL) model (NASA 2009)

INCOSE SE Competency Model

The INCOSE model was developed by a working group in the United Kingdom (Cowper et al. 2005). As Table 2 shows, the INCOSE framework is divided into three theme areas - systems thinking, holistic life cycle view, and systems management - with a number of competencies in each. The INCOSE UK model was later adopted by the broader INCOSE organization (INCOSE 2010).

Table 2. INCOSE UK Working Group Competency (INCOSE UK 2010).

Systems Thinking	System Concepts	
	Super-System Capability Issues	
	Enterprise and Technology Environment	
Hollistic Lifecycle View	Determining and Managing Stakeholder Requirements	
	Systems Design	Architectural Design
		Concept Generation
		Design For...
		Functional Analysis
		Interface Management
		Maintain Design Integrity
		Modeling and Simulation
		Select Preferred Solution
		System Robustness
	Systems Intergration & Verification	
	Validation	
	Transition to Operation	
Systems Engineering Management	Concurrent Engineering	
	Enterprise Integration	
	Integration of Specialties	
	Lifecycle Process Definition	
	Planning, Monitoring, and Controlling	

United States DoD Engineering Competency Model

The model for US Department of Defense (DoD) acquisition engineering professionals (ENG) includes 41 competency areas, as shown in Table 3 (DAU 2013). Each is grouped according to a “Unit of Competence” as listed in the left-hand column. For this model, the four top-level groupings are analytical, technical management, professional, and business acumen. The life cycle view used in the INCOSE model is evident in the ENG analytical grouping, but is not cited explicitly. Technical management is the equivalent of the INCOSE SE management, but additional competencies are added, including software engineering competencies and acquisition. Selected general professional skills have been added to meet the needs for strong leadership required of the acquisition engineering professionals. The business acumen competencies were added to meet the needs of these professionals to be able to support contract development and oversight activities and to engage with the defense industry.

Table 3. DoD Competency Model (DAU 2013) Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Analytical (11)	1. Mission-Level Assessment
	2. Stakeholder Requirements Definition
	3. Requirements Analysis
	4. Architecture Design
	5. Implementation
	6. Intergration
	7. Verification
	8. Validation
	9. Transition
	10. Design Considerations
	11. Tools and Techniques
Technical Management (10)	12. Decision Analysis
	13. Technical Planning
	14. Technical Assessment
	15. Configuration Management
	16. Requirements Management
	17. Risk Management
	18. Data Management
	19. Interface Management
	20. Software Engineering
	21. Acquisition

Professional (10)	22. Problem Solving
	23. Strategic Thinking
	24. Professional Ethics
	25. Leading High-Performance Teams
	26. Communication
	27. Coaching and Mentoring
	28. Managing Stakeholders
	29. Mission and Results Focus
	30. Personal Effectiveness/Peer Interaction
	31. Sound Judgment
Business Acumen (10)	32. Industry Landscape
	33. Organization
	34. Cost, Pricing, and Rates
	35. Cost Estimating
	36. Financial Reporting and Metrics
	37. Business Strategy
	38. Capture Planning and Proposal Process
	39. Supplier Management
	40. Industry Motivation, Incentives, Rewards
	41. Negotiations

NASA SE Competency Model

The US National Aeronautics and Space Administration (NASA) APPEL website provides a competency model that covers both project engineering and systems engineering (APPEL 2009). There are three parts to the model, one that is unique to project engineering, one that is unique to systems engineering, and a third that is common to both disciplines. Table 4 below shows the SE aspects of the model. The project management items include project conceptualization, resource management, project implementation, project closeout, and program control and evaluation. The common competency areas are NASA internal and external environments, human capital and management, security, safety and mission assurance, professional and leadership development, and knowledge management. This 2010 model is adapted from earlier versions. (Squires at al. 2010, 246-260) offer a method that can be used to analyze the degree to which an organization’s SE capabilities meet government-industry defined SE needs.

Table 4. SE Portion of the APPEL Competency Model (APPEL 2009). Released by NASA APPEL.

System Design	SE 1.1 - Stakeholder Expectation Definition & Management
	SE 1.2 - Technical Requirements Definition
	SE 1.3 - Logical Decomposition
	SE 1.4 - Design Solution Definition
Product Realization	SE 2.1 - Product Implementation
	SE 2.2 - Product Integration
	SE 2.3 - Product Verification
	SE 2.4 - Product Validation
	SE 2.5 - Product Transition
Technical Management	SE 3.1 - Technical Planning
	SE 3.2 - Requirements Management
	SE 3.3 - Interface Management
	SE 3.4 - Technical Risk Management
	SE 3.5 - Configuration Management
	SE 3.6 - Technical Data Management
	SE 3.7 - Technical Assessment
	SE 3.8 - Technical Decision Analysis

Relationship of SE Competencies to Other Competencies

SE is one of many engineering disciplines. A competent SE must possess KSAAs that are unique to SE, as well as many other KSAAs that are shared with other engineering and non-engineering disciplines.

One approach for a complete engineering competency model framework has multiple dimensions where each of the dimensions has unique KSAAs that are independent of the other dimensions (Wells 2008). The number of dimensions depends on the engineering organization and the range of work performed within the organization. The concept of creating independent axes for the competencies was presented in Jansma and Derro (2007), using technical knowledge (domain/discipline specific), personal behaviors, and process as the three axes. An approach that uses process as a dimension is presented in Widmann et al. (2000), where the competencies are mapped to process and process maturity models. For a large engineering organization that creates complex systems solutions, there are typically four dimensions:

1. **Discipline** (e.g., electrical, mechanical, chemical, systems, optical);
2. **Life Cycle** (e.g., requirements, design, testing);
3. **Domain** (e.g., aerospace, ships, health, transportation); and
4. **Mission** (e.g., air defense, naval warfare, rail transportation, border control, environmental protection).

These four dimensions are built on the concept defined in Jansma and Derro (2007) and Widmann et al. (2000) by separating discipline from domain and by adding mission and life cycle dimensions. Within many organizations, the mission may be consistent across the organization and this dimension would be unnecessary. A three-dimensional example is shown in Figure 1, where the organization works on only one mission area so that dimension has been eliminated from the framework.

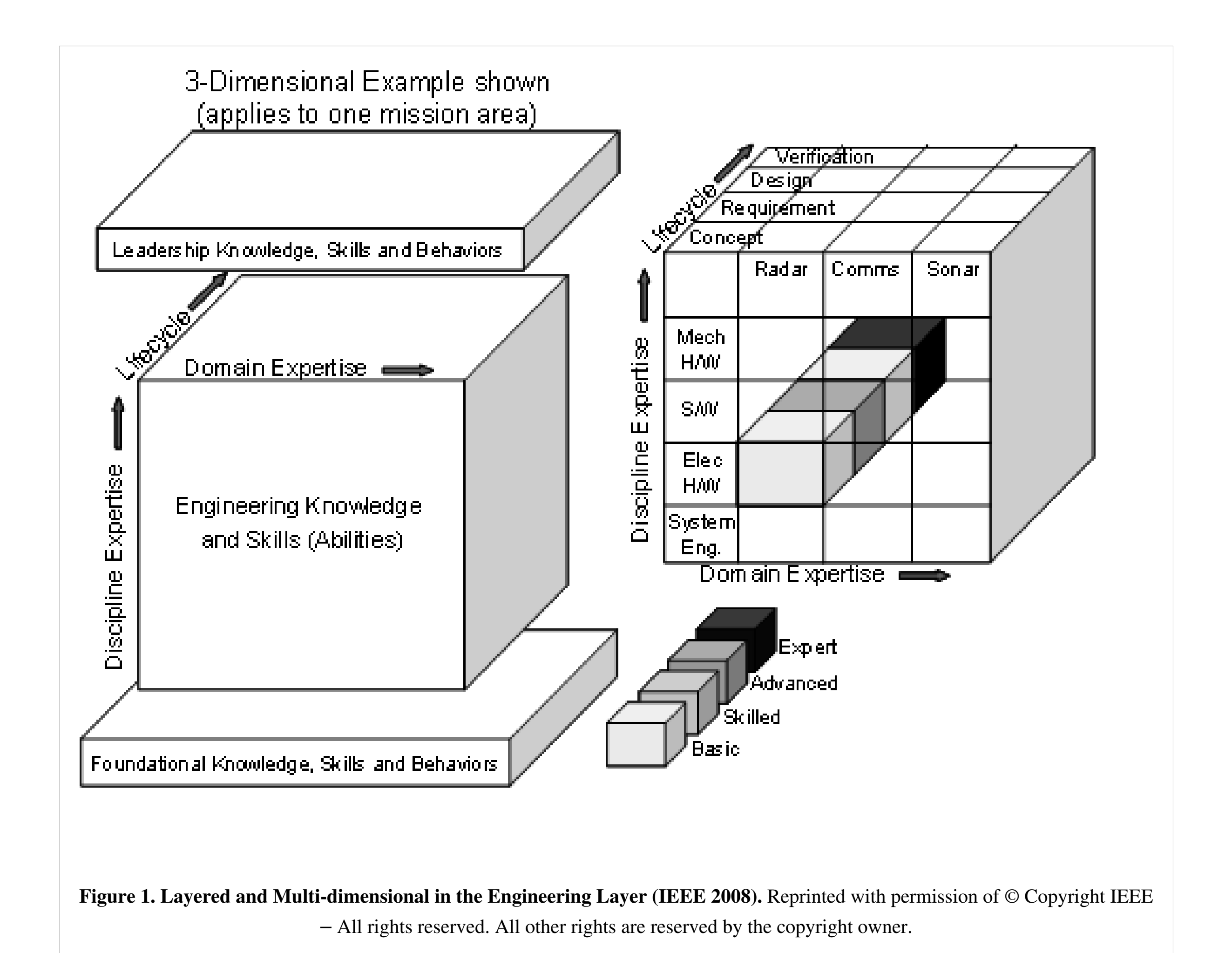


Figure 1. Layered and Multi-dimensional in the Engineering Layer (IEEE 2008). Reprinted with permission of © Copyright IEEE – All rights reserved. All other rights are reserved by the copyright owner.

The discipline, domain, and life cycle dimensions are included in this example, and some of the first-level areas in each of these dimensions are shown. At this level, an organization or an individual can indicate which areas are included in their existing or desired competencies. The sub-cubes are filled in by indicating the level of proficiency that exists or is required. For this example, blank indicates that the area is not applicable, and colors (shades of gray) are used to indicate the levels of expertise. The example shows a radar electrical designer that is an expert at hardware verification, is skilled at writing radar electrical requirements, and has some knowledge of electrical hardware concepts and detailed design. The radar electrical designer would also assess his or her proficiency in the other areas, the foundation layer, and the leadership layer to provide a complete assessment.

References

Works Cited

Armstrong, J.R., D. Henry, K. Kepcher, and A. Pyster. 2011. "Competencies Required for Successful Acquisition of Large, Highly Complex Systems of Systems." Paper presented at 21st Annual International Council on Systems Engineering (INCOSE) International Symposium (IS), 20-23 June 2011, Denver, CO, USA.

Bloom, Benjamin S., Max D. Engelhart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. 1956. *Taxonomy of Educational Objectives.* New York: David McKay.

Cowper, D., S. Bennison, R. Allen-Shalless, K. Barnwell, S. Brown, A. El Fatatry, J. Hooper, S. Hudson, L. Oliver, and A. Smith. 2005. *Systems Engineering Core Competencies Framework*. Folkestone, UK: International Council on Systems Engineering (INCOSE) UK Advisory Board (UKAB).

DAU. 2013. *ENG Competency Model*, 12 June 2013 version. in Defense Acquisition University (DAU)/U.S. Department of Defense Database Online. Accessed on June 3, 2015. Available at https://dap.dau.mil/workforce/Documents/Comp/ENG%20Competency%20Model%2020130612_Final.pdf.

ETA. 2010. *Career One Stop: Competency Model Clearing House: Aerospace Competency Model.* in Employment and Training Administration (ETA)/U.S. Department of Labor. Washington, DC. Accessed on September 15, 2011. Available at http://www.careeronestop.org//competencymodel/pyramid.aspx?AEO=Y.

Ferris, T.L.J. 2010. "Comparison of Systems Engineering Competency Frameworks." Paper presented at the 4th Asia-Pacific Conference on Systems Engineering (APCOSE), Systems Engineering: Collaboration for Intelligent Systems, 3-6 October 2010, Keelung, Taiwan.

Frank, M. 2000. "Engineering Systems Thinking and Systems Thinking." *Systems Engineering.* 3(3): 163-168.

Frank, M. 2002. "Characteristics of Engineering Systems Thinking – A 3-D Approach for Curriculum Content." *IEEE Transaction on System, Man, and Cybernetics.* 32(3) Part C: 203-214.

Frank, M. 2006. "Knowledge, Abilities, Cognitive Characteristics and Behavioral Competences of Engineers with High Capacity for Engineering Systems Thinking (CEST)." *Systems Engineering.* 9(2): 91-103. (Republished in *IEEE Engineering Management Review.* 34(3)(2006):48-61).

Hall, A.D. 1962. *A Methodology for Systems Engineering*. Princeton, NJ, USA: D. Van Nostrand Company Inc.

Holt, Jon, and Perry, Simon,*A Pragmatic Guide to Competency, Tools, Frameworks, and Assessment.*BCS, The Chartered Institute for IT, Swindon, UK, 2011.

INCOSE. 2011. "History of INCOSE Certification Program." Accessed April 13, 2015 at http://www.incose.org/certification/CertHistory

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205.* San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

INCOSE UK. 2010. "Systems Engineering Competency Framework," Accessed on June 3, 2015. Available at, <http://www.incoseonline.org.uk/Normal_Files/Publications/Framework.aspx?CatID=Publications&SubCat=INCOSEPublications>>

Jansma, P.A. and M.E. Derro. 2007. "If You Want Good Systems Engineers, Sometimes You Have to Grow Your Own!." Paper presented at IEEE Aerospace Conference. 3-10 March, 2007. Big Sky, MT, USA.

Kasser, J.E., D. Hitchins, and T.V. Huynh. 2009. "Reengineering Systems Engineering." Paper presented at the 3rd Annual Asia-Pacific Conference on Systems Engineering (APCOSE), 2009, Singapore.

Krathwohl, David. *A Revision of Bloom's Taxonomy: An Overview.* 2002.

Menrad, R. and H. Lawson. 2008. "Development of a NASA Integrated Technical Workforce Career Development Model Entitled: Requisite Occupation Competencies and Knowledge – The ROCK." Paper presented at the 59th International Astronautical Congress (IAC), 29 September-3 October, 2008, Glasgow, Scotland.

MITRE. 2007. "MITRE Systems Engineering (SE) Competency Model." Version 1.13E. September 2007. Accessed on June 3, 2015. Available at, http://www.mitre.org/publications/technical-papers/systems-engineering-competency-model.

NASA. 2009. *NASA Competency Management Systems (CMS): Workforce Competency Dictionary,* revision 7a. U.S. National Aeronautics and Space Administration (NASA). Washington, D.C.

NASA. 2009. *Project Management and Systems Engineering Competency Model.* Academy of Program/Project & Engineering Leadership (APPEL). Washington, DC, USA: US National Aeronautics and Space Administration (NASA). Accessed on June 3, 2015. Available at http://appel.nasa.gov/competency-model/.

SEI. 2007. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.2, Measurement and Analysis Process Area. Pittsburg, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

SEI. 2004. *CMMI-Based Professional Certifications: The Competency Lifecycle Framework,* Software Engineering Institute, CMU/SEI-2004-SR-013. Accessed on June 3, 2015. Available at http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6833.

Squires, A., W. Larson, and B. Sauser. 2010. "Mapping space-based systems engineering curriculum to government-industry vetted competencies for improved organizational performance." *Systems Engineering*. 13 (3): 246-260. Available at, http://dx.doi.org/10.1002/sys.20146.

Squires, A., J. Wade, P. Dominick, and D. Gelosh. 2011. "Building a Competency Taxonomy to Guide Experience Acceleration of Lead Program Systems Engineers." Paper presented at the Conference on Systems Engineering Research (CSER), 15-16 April 2011, Los Angeles, CA.

Wells, B.H. 2008. "A Multi-Dimensional Hierarchical Engineering Competency Model Framework." Paper presented at IEEE International Systems Conference, March 2008, Montreal, Canada. Whitcomb, Clifford, Rabia Khan, and Corina White. 2014. "Systems Engineering Competency FY14 Technical Report." Naval Postgraduate School Technical Report, Monterey, CA. https://calhoun.nps.edu/handle/10945/44705

Whitcomb, Clifford, Leslie Whitcomb. 2013. Effective Interpersonal and Team Communication Skills for Engineers. IEEE Press, John Wiley and Sons.

Whitcomb, Clifford, Rabia Khan, and Corina White. 2013. "Systems Engineering Competency FY13 Technical Report." Naval Postgraduate School Technical Report, Monterey, CA. Accessed on June 4, 2015. Available at, https://calhoun.nps.edu/handle/10945/43424.

Whitcomb, Clifford, Rabia Khan, and Corina White. 2014. "Systems Engineering Competency FY14 Technical Report." Naval Postgraduate School Technical Report, Monterey, CA. Accessed on June 4, 2015. Available at, https://calhoun.nps.edu/handle/10945/44705.

Widmann, E.R., G.E. Anderson, G.J. Hudak, and T.A. Hudak. 2000. "The Taxonomy of Systems Engineering Competency for The New Millennium." Presented at 10th Annual INCOSE Internal Symposium, 16-20 July 2000, Minneapolis, MN, USA.

Primary References

DAU. 2013. *ENG Competency Model*, 12 June 2013 version. in Defense Acquisition University (DAU)/U.S. Department of Defense Database Online. Accessed on June 3, 2015. Available at https://dap.dau.mil/workforce/Documents/Comp/ENG%20Competency%20Model%2020130612_Final.pdf.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205.* San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

Additional References

Whitcomb, Clifford, Jessica Delgado, Rabia Khan, Juli Alexander, Corina White, Dana Grambow, Paul Walter. 2015. "The Department of the Navy Systems Engineering Career Competency Model." Proceedings of the Twelfth Annual Acquisition Research Symposium. Naval Postgraduate School, Monterey, CA..

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Assessing Individuals

The ability to fairly assess individuals is a critical aspect of enabling individuals. This article describes how to assess the systems engineering (glossary) (SE) competencies needed and possessed by an individual, as well as that individual’s SE performance.

Assessing Competency Needs

If an organization wants to use its own customized competency model, an initial decision is *make vs. buy.* If there is an existing SE competency model that fits the organization's context and purpose, the organization might want to use the existing SE competency model directly. If existing models must be tailored or a new SE competency model developed, the organization should first understand its context.

Determining Context

Prior to understanding what SE competencies are needed, it is important for an organization to examine the situation in which it is embedded, including environment, history, and strategy. As Figure 1 shows, MITRE has developed a framework characterizing different levels of systems complexity. (MITRE 2007, 1-12) This framework may help an organization identify which competencies are needed. An organization working primarily in the *traditional program domain* may need to emphasize a different set of competencies than an organization working primarily in the *messy frontier.* If an organization seeks to improve existing capabilities in one area, extensive technical knowledge in that specific area might be very important. For example, if stakeholder involvement is characterized by multiple equities and distrust, rather than collaboration and concurrence, a higher level of competency in being able to balance stakeholder requirements might be needed. If the organization's desired outcome builds a fundamentally new capability, technical knowledge in a broader set of areas might be useful.

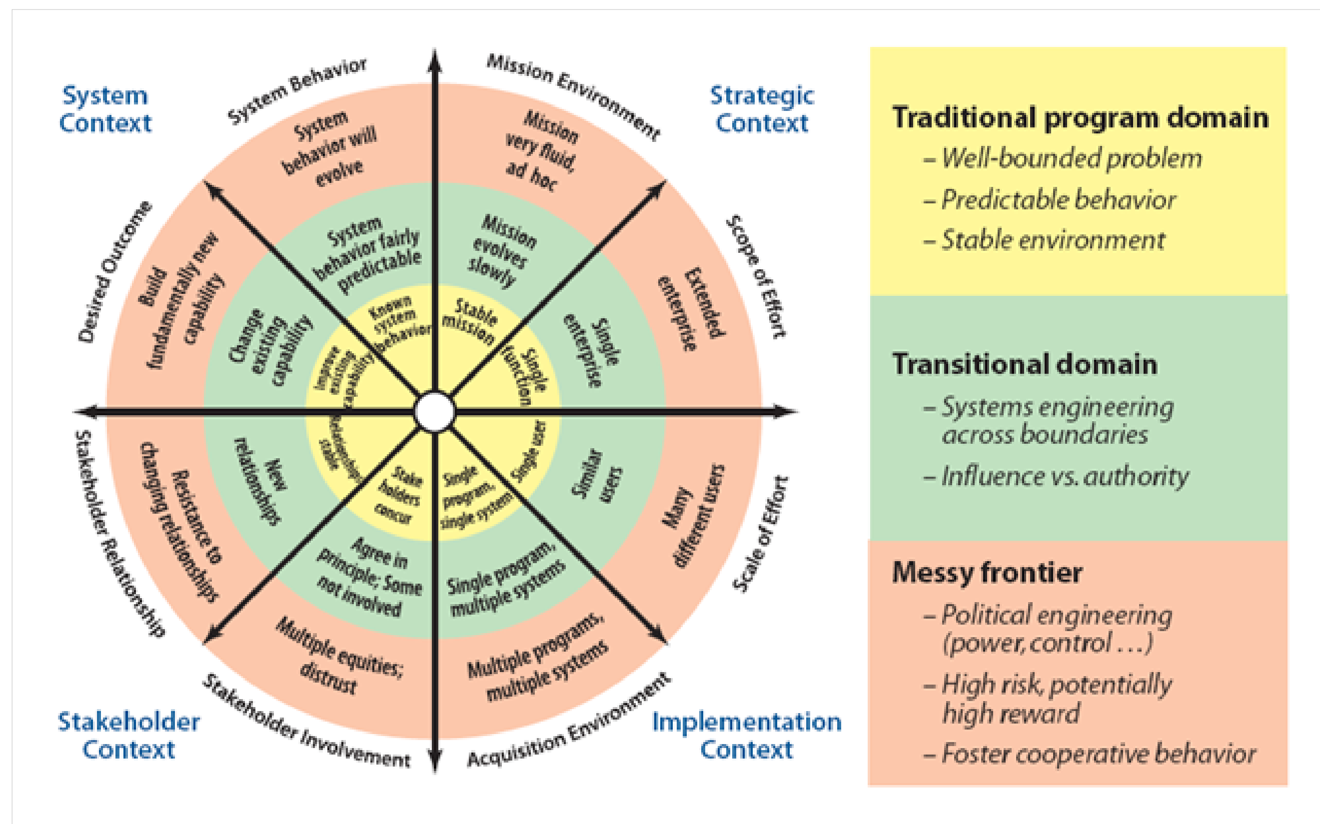


Figure 1. MITRE Enterprise Systems Engineering Framework (MITRE 2007). Reprinted with permission of © 2011. The MITRE Corporation. All Rights Reserved. All other rights are reserved by the copyright owner.

In addition, an organization might consider both its current situation and its forward strategy. For example, if an organization has previously worked in a traditional systems engineering context (MITRE 2007) but has a strategy to transition into enterprise systems engineering (ESE) work in the future, that organization might want to develop a competency model both for what was important in the traditional SE context and for what will be required for ESE work. This would also hold true for an organization moving to a different contracting environment where competencies, such as the ability to properly tailor the SE approach to *right size* the SE effort and balance cost and risk, might be more important.

Determining Roles and Competencies

Once an organization has characterized its context, the next step is to understand exactly what SE roles are needed and how those roles will be allocated to teams and individuals. To assess the performance of an individual, it is essential to explicitly state the roles and competencies required for that individual. See the references in Roles and Competencies for guides to existing SE standards and SE competency models.

Assessing Individual SE Competency

In order to demonstrate competence, there must be some way to qualify and measure it, and this is where competency assessment is used (Holt and Perry 2011). This assessment informs the interventions needed to further develop individual SE KSAA upon which competency is based. Described below are possible methods which may be used for assessing an individual's current competency level; an organization should choose the correct model based on their context, as identified previously.

Proficiency Levels

In order to provide a context for individuals and organizations to develop competencies, a consistent system of defining KSAAs should be created. One popular method is based on Bloom’s taxonomy (Bloom 1984), presented below for the cognitive domain in order from least complex to most complex cognitive ability.

- **Remember**: Recall or recognize terms, definitions, facts, ideas, materials, patterns, sequences, methods, principles, etc.
- **Understand**: Read and understand descriptions, communications, reports, tables, diagrams, directions, regulations, etc.
- **Apply**: Know when and how to use ideas, procedures, methods, formulas, principles, theories, etc.
- **Analyze**: Break down information into its constituent parts and recognize their relationship to one another and how they are organized; identify sublevel factors or salient data from a complex scenario.
- **Evaluate**: Make judgments about the value of proposed ideas, solutions, etc., by comparing the proposal to specific criteria or standards.
- **Create**: Put parts or elements together in such a way as to reveal a pattern or structure not clearly there before; identify which data or information from a complex set is appropriate to examine further or from which supported conclusions can be drawn.

One way to assess competency is to assign KSAAs to proficiency level categories within each competency. Examples of proficiency levels include the INCOSE competency model, with proficiency levels of: awareness, supervised practitioner, practitioner, and expert (INCOSE 2010). The Academy of Program/Project & Engineering Leadership (APPEL) competency model includes the levels: participate, apply, manage, and guide, respectively (Menrad and Lawson 2008). The U.S. National Aeronautics and Space Administration (NASA), as part of the APPEL (APPEL 2009), has also defined proficiency levels: technical engineer/project team member, subsystem lead/manager, project manager/project systems engineer, and program manager/program systems engineer. The Defense Civilian Personnel Advisory Service (DCPAS) defines a 5-tier framework to indicate the degree to which employees perform competecnies as awareness, basic, intermediate, advanced, and expert.

The KSAAs defined in the lower levels of the cognitive domain (remember, understand) are typically foundational, and involve demonstration of basic knowledge. The higher levels (apply, analyze, evaluate, and create) reflect higher cognitive ability. Cognitive and affective processes within Bloom's taxonomy refer to levels of observable actions that indicate learning is occuring (Whitcomb, Delgado, Khan, Alexander, White, Grambow, Walter 2015). The Bloom's domain levels should not be used exclusively to determine the proficiency levels required for attainment or assessment of a competency. Higher level cognitive capabilites belong across proficiency levels, and should be used as appropriate to the KSAA involved. These higher level terms infer some observbable action or outcome, so the context for assessing the attainment of the KSAA, or a group of KSAAs, related to a competency needs to be defined. For example, applying SE methods can be accomplished on simple subsystems or systems and so perhaps belong in a lower proficiency level such as supervised practitioner. Applying SE methods to complex enterprise or systems of systems, may belong in the practitioner or even the expert level. The determination of what proficiency level is desired for each KSAA is determined by the organization, and may vary among different organizations.

Quality of Competency Assessment

When using application as a measure of competency, it is important to have a measure of *goodness*. If someone is applying a competency in an exceptionally complex situation, they may not necessarily be successful in this application. An individual may be *managing and guiding*, but this is only helpful to the organization if it is being done well. In addition, an individual might be fully proficient in a particular competency, but not be given an opportunity to use that competency; for this reason, it is important to understand the context in which these competencies are being assessed.

Individual SE Competency versus Performance

Even when an individual is highly proficient in an SE competency, context may preclude exemplary performance of that competency. For example, an individual with high competency in risk management may be embedded in a team or an organization which ignores that talent, whether because of flawed procedures or some other reason. Developing individual competencies is not enough to ensure exemplary SE performance.

When SE roles are clearly defined, performance assessment at least has a chance to be objective. However, since teams are most often tasked with accomplishing the SE tasks on a project, it is the team's performance which ends up being assessed. (See Team Capability). The final execution and performance of SE is a function of competency, capability, and capacity. (See Enabling Teams and Enabling Businesses and Enterprises.)

References

Works Cited

Academy of Program/Project & Engineering Leadership (APPEL). 2009. NASA's Systems Engineering Competencies. Washington, DC, USA: U.S. National Aeronautics and Space Administration. Available at: http://www.nasa.gov/offices/oce/appel/pm-development/pm_se_competency_framework.html.

Bloom, B. S. 1984. *Taxonomy of Educational Objectives.* New York, NY, USA: Longman.

Holt, Jon, and Simon Perry. 2011. *A Pragmatic Guide to Competency, Tools, Frameworks, and Assessment.* BCS, The Chartered Institute for IT, Swindon, UK.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205*. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

Menrad, R. and H. Lawson. 2008. "Development of a NASA Integrated Technical Workforce Career Development Model Entitled: Requisite Occupation Competencies and Knowledge – The ROCK." Paper presented at the 59th International Astronautical Congress (IAC). 29 September-3 October 2008. Glasgow, Scotland.

MITRE. 2007. *Enterprise Architecting for Enterprise Systems Engineering*. Warrendale, PA, USA: SEPO Collaborations, SAE International. June 2007.

Whitcomb, Clifford, Jessica Delgado, Rabia Khan, Juli Alexander, Corina White, Dana Grambow, Paul Walter. 2015. "The Department of the Navy Systems Engineering Career Competency Model." Proceedings of the Twelfth Annual Acquisition Research Symposium. Naval Postgraduate School, Monterey, CA.

Primary References

Academy of Program/Project & Engineering Leadership (APPEL). 2009. *NASA's Systems Engineering Competencies.* Washington, DC, USA: U.S. National Aeronautics and Space Administration (NASA). Accessed on May 2, 2014. Available at http:/ / appel. nasa. gov/ career-resources/ project-management-and-systems-engineering-competency-model/.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205.* San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

Additional References

Holt, J., and S. Perry. 2011. *A pragmatic guide to competency: tools, frameworks and assessment*. Swindon, UK: British Computer Society.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Developing Individuals

Developing each individual’s systems engineering (SE) competencies(glossary) is a key aspect of enabling individuals. The goal may be to develop competency in a broad range of SE competencies or a single aspect of SE, and it is important to know exactly which SE competencies are desired. This article describes strategies to develop SE competencies in individuals.

Closing Competency Gaps

Delivering excellent systems that fulfill customer needs is the primary goal of the organization. Developing *the capability* to deliver such systems is a secondary goal, and while necessary, is not sufficient. To attain both of these goals, the organization must assess itself and effect a strategy to identify and close competency gaps.

To identify competency gaps, an organization may take two basic steps:

1. Listing desired competencies, as discussed in Roles and Competencies; and
2. Assessing the competencies of individual systems engineers, as discussed in Assessing Individuals.

Models useful for listing competencies include the International Council on Systems Engineering (INCOSE) United Kingdom Advisory Board model (Cowper et al. 2005; INCOSE 2010), the ENG Competency Model (DAU 2013), and the Academy of Program/Project & Engineering Leadership (APPEL 2009) model (Menrad and Lawson 2008).

Once the organization knows the SE competencies it needs to develop to close the competency gaps it has identified, it may choose from the several methods (Davidz and Martin 2011) outlined in the table below.

Table 1. SE Competency Development Framework. (SEBoK Original)

Goal	Objective	Method
PRIMARY GOAL = Delivery of excellent systems to fulfill customer needs	Focus on successful performance outcome	Corporate intiatives
	Focus on performance of project team	Team coaching of project team for performance enhancement
SECONDARY GOAL = Competency to deliver excellent systems to fulfill customer needs	Develop individual competency	Training courses
		Job rotation
		Mentoring
		Hands-on experience
		Develop a few hand-picked individuals
		University educational degree program
		Customized educational program
		Combination program - education, training, job rotation, mentoring, hands-on experience
		Course certificate program
	Ensure individual competency through certification	Certification program
	Filter those working in systems roles	Use individual characteristics to select employees for systems roles
	Ensure organizational competency through certification	ISO 9000
	Develop organizational systems competency through processes	Process improvement using an established framework
		Concept maps to identify the thought processes of senior systems engineers
		Standarize systems policies and procedures for consistency
		Systems engineering web portal
		Systems knowledge management repository
		On-call organizational experts
		Rotating professor who works at company part-time and is at university part-time

System Delivery

Some organizations mount initiatives which focus directly on successful system delivery. Others focus on project team performance, in some cases by offering coaching, as a means to ensure successful system delivery.

One example of the latter approach is the performance enhancement service of the US National Aeronautics and Space Administration (NASA) Academy of Program/Project & Engineering Leadership (APPEL), which assesses team performance and then offers developmental interventions with coaching (NASA 2010).

Organizations pursue multiple paths towards developing the capability to deliver excellent systems, including

- developing the competency of individuals;
- developing the competency of the organization through processes (Davidz and Maier 2007); and
- putting measures should in place to verify the efficacy of the selected methods.

Individual Competency

An organization may choose a combination of methods to develop individual systems competency. General Electric's Edison Engineering Development Program (GE 2010) and Lockheed Martin's Leadership Development Programs (Lockheed Martin 2010) are examples among the many combination programs offered within companies.

Whether or not the program is specifically oriented to develop systems skills, the breadth of technical training and experience, coupled with business training, can produce a rich understanding of systems for the participant. Furthermore, new combination programs can be designed to develop specific systems-oriented skills for an organization.

Methods for developing individual competency include

- **classroom or online training courses**, a traditional choice for knowledge transfer and skill acquisition. Here, an instructor directs a classroom of participants. The method of instruction may vary from a lecture format to case study work to hands-on exercises. The impact and effectiveness of this method varies considerably based on the skill of the instructor, the effort of the participants, the presentation of the material, the course content, the quality of the course design process, and the matching of the course material to organizational needs. These types of interventions may also be given online. Squires (2011) investigates the relationship between online pedagogy and student perceived learning of SE competencies.
- **job rotation**, where a participant rotates through a series of work assignments that cut across different aspects of the organization to gain broad experience in a relatively short time.
- **mentoring**, where a more experienced individual is paired with a protégé in a developmental relationship. Many organizations use mentoring, whose impact and effectiveness vary considerably. Success factors are the tenable pairing of individuals, and the provision of adequate time for mentoring.
- **hands-on experience**, where organizations provide for their engineers to get hands-on experience that they would otherwise lack. A research study by Davidz on enablers and barriers to the development of systems thinking showed that systems thinking is developed primarily by experiential learning (Davidz 2006; Davidz and Nightingale 2008, 1-14). As an example, some individuals found that working in a job that dealt with the full system, such as working in an integration and test environment, enabled development of systems thinking.
- **selecting individuals** who appear to have high potential and focusing on their development. Hand-selection may or may not be accompanied by the other identified methods.
- **formal education**, such as a university degree program. A growing number of SE degree programs are offered worldwide (Lasfer and Pyster 2011). Companies have also worked with local universities to set up customized educational programs for their employees. The company benefits because it can tailor the educational program to the unique needs of its business. In a certificate program, individuals receive a certificate for taking a specific set of courses, either at a university or as provided by the company. There are a growing number of certificate programs for developing systems competency.

Individual Certification

Organizations may seek to boost individual systems competency through certification programs. These can combine work experience, educational background, and training classes. Certifications are offered by local, national, and international professional bodies.

SE organizations may encourage employees to seek certification from the International Council on Systems Engineering (INCOSE 2011) or may use this type of certification as a filter (see **Filters**, below). In addition, many companies have developed their own internal certification measures. For example, the Aerospace Corporation has an Aerospace Systems Architecting and Engineering Certificate Program (ASAECP). (Gardner 2007)

Filters

Another approach to developing individual competency is to select employees for systems roles based on certain characteristics, or filters. Before using a list of characteristics for filtering, though, an organization should critically examine

1. how the list of individual characteristics was determined, and
2. how the characteristics identified enable the performance of a systems job.

Characteristics used as filters should

- enable one to perform a systems job
- be viewed as important to perform a systems job, or
- be necessary to perform a systems job.

A necessary characteristic is much stronger than an enabling one, and before filtering for certain traits, it is important to understand whether the characteristic is an enabler or a necessity.

Finally, it is important to understand the extent to which findings are generally applicable, since a list of characteristics that determine success in one organization may not be generalizable to another organization.

Organizational Capability

Once an organization has determined which SE capabilities are mission critical (please see Deciding on Desired Systems Engineering Capabilities within Businesses and Enterprises), there are many different ways in which an organization can seek to develop or improve these capabilities. Some approaches seen in the literature include the following:

- Organizations may choose to develop organizational systems capability through processes. One method organizations may choose is to pursue process improvement using an established framework. An example is the Capability Maturity Model® Integration (CMMI) process improvement approach (SEI 2010, 1).
- Concept maps - graphical representations of engineering thought processes - have been shown to be an effective method of transferring knowledge from senior engineering personnel to junior engineering personnel (Kramer 2007, 26-29; Kramer 2005). These maps may provide a mechanism for increasing knowledge of the systems engineering population of an organization.
- An organization may also choose to develop organizational systems competencies by standardizing systems policies and procedures. An example from NASA is their *NASA Systems Engineering Processes and Requirement* (NASA 2007).
- Some organizations use a web portal to store and organize applicable systems engineering knowledge and processes, which assists in developing organizational systems competency. An example is the Mission Assurance Portal for the Aerospace Corporation (Roberts et al. 2007, 10-13).
- Another approach being considered in the community is the development of a rotating professor role, where the person would work at the company and then be at a university to strengthen the link between academia and industry.
- Another approach is to alter organizational design to foster and mature a desired competency. For example, an organization that identifies competency in the area of reliability as critical to its SE success may develop a reliability group, which will help foster growth and improvement in reliability competencies.

Organizational Certification

Certification at the organizational level exists also, and can be a means for ensuring competency. ISO certification is one example (ISO 2010). Before taking this approach, the organization should verify that the capabilities required by the certification are indeed the systems capabilities it seeks. For more on determining appropriate organizational capabilities, see Deciding on Desired Systems Engineering Capabilities within Businesses and Enterprises.

Repositioning the Product Life Cycle

An organization may also choose to reposition its product life cycle philosophy to maintain system competency. For example, NASA has done this with its APPEL program (APPEL 2009).

Since the systems competencies of individuals are primarily developed through experiential learning, providing experiential learning opportunities is critical. Shortening the product life cycle is one way to ensure that individuals acquire the full range of desired competency sooner.

Maintaining Competency Plans

An organization that has developed an SE competency plan should consider how to maintain it. How, and how often, will the competency plan be re-examined and updated? The maintenance process should account for the ongoing evolution of global contexts, business strategies, and the SEBoK. The process for assessing competencies and taking action to improve them must be part of the normal operations of the organization and should occur periodically.

References

Works Cited

Academy of Program/Project & Engineering Leadership (APPEL). 2009. NASA's Systems Engineering Competencies. Washington, D.C.: U.S. National Aeronautics and Space Association. Accessed on September 15, 2011. Available at http://www.nasa.gov/offices/oce/appel/pm-development/pm_se_competency_framework.html.

Cowper, D., S. Bennison, R. Allen-Shalless, K. Barnwell, S. Brown, A. El Fatatry, J. Hooper, S. Hudson, L. Oliver, and A. Smith. 2005. *Systems Engineering Core Competencies Framework.* Folkestone, UK: International Council on Systems Engineering (INCOSE) UK Advisory Board (UKAB).

Davidz, H.L. and J. Martin. 2011. "Defining a Strategy for Development of Systems Capability in the Workforce". *Systems Engineering*. 14(2): 141-143.

Davidz, H.L. and M.W. Maier. 2007. "An Integrated Approach to Developing Systems Professionals." Paper presented at the 17th Annual International Council on Systems Engineering (INCOSE) International Symposium, 24-28 June 2007. San Diego, CA, USA.

Davidz, H.L., and D. Nightingale. 2008. "Enabling Systems Thinking to Accelerate the Development of Senior Systems Engineers." *Systems Engineering*. 11(1): 1-14.

Davidz, H.L. 2006. *Enabling Systems Thinking to Accelerate the Development of Senior Systems Engineers.* Dissertation. Massachusetts Institute of Technology (MIT), Cambridge, MA, USA.

Gardner, B. 2007. "A Corporate Approach to National Security Space Education." *Crosslink,* the Aerospace Corporation Magazine of Advances in Aerospace Technology. 8(1) (Spring 2007):10-5. Accessed April 23, 2013. Available at: http://aerospace.wpengine.netdna-cdn.com/wp-content/uploads/crosslink/V8N1.pdf.

GE. 2010. *Edison Engineering Development Program (EEDP) in General Electric.* Accessed on September 15, 2011. Available at http://www.gecareers.com/GECAREERS/jsp/us/studentOpportunities/leadershipPrograms/eng_program_guide.jsp.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205.* San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

INCOSE. 2011. "Systems Engineering Professional Certification." In *International Council on Systems Engineering* online. Accessed April 13, 2015. Available at: http://www.incose.org/certification/.

Kramer, M.J. 2007. "Can Concept Maps Bridge The Engineering Gap?" *Crosslink*, the Aerospace Corporation Magazine of Advances in Aerospace Technology. 8(1) (Spring 2007): 26-9. Accessed April 23, 2013. Available at: http://aerospace.wpengine.netdna-cdn.com/wp-content/uploads/crosslink/V8N1.pdf.

Kramer, M.J. 2005. *Using Concept Maps for Knowledge Acquisition in Satellite Design: Translating 'Statement of Requirements on Orbit' to 'Design Requirements.* Dissertation. Ft. Lauderdale, FL, USA: Graduate School of Computer and Information Sciences, Nova Southeastern University.

Lasfer, K. and A. Pyster. 2011. "The Growth of Systems Engineering Graduate Programs in the United States." Paper presented at Conference on Systems Engineering Research, 15-16 April 2011. Los Angeles, CA, USA.

Lockheed Martin. 2010. *Training and Leadership Development Programs for College Applicants in Lockheed Martin Corporation.* Bethesda, MD, USA. Accessed on August 30, 2012. Available at http:/ / www. lockheedmartinjobs.com/leadership-development-program.asp.

NASA. 2010. *Academy of Program/Project & engineering leadership (APPEL): Project life cycle support in U.S. National Aeronautics and Space Administration (NASA).* Washington, DC, USA: U.S. National Air and Space Administration (NASA). Accessed on September 15, 2011. Available at http://www.nasa.gov/offices/oce/appel/ performance/lifecycle/161.html.

NASA. 2007. *NASA Procedural Requirements: NASA Systems Engineering Processes and Requirements*. Washington, DC, USA: U.S. National Aeronautic and Space Administration (NASA). NPR 7123.1A.

Roberts, J., B. Simpson, and S. Guarro. 2007. "A Mission Assurance Toolbox." *Crosslink*, the Aerospace Corporation Magazine of Advances in Aerospace Technology. 8(2) (Fall 2007): 10-13.

SEI. 2007. *Capability Maturity Model Integrated (CMMI) for Development*, version 1.2, Measurement and Analysis Process Area. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Squires, A. 2011. *Investigating the Relationship between Online Pedagogy and Student Perceived Learning of Systems Engineering Competencies*. Dissertation. Stevens Institute of Technology, Hoboken, NJ, USA.

Primary References

Academy of Program/Project & Engineering Leadership (APPEL). 2009. *NASA's Systems Engineering Competencies*. Washington, DC, USA: U.S. National Aeronautics and Space Administration (NASA). Accessed on May 2, 2014. Available at http:/ / appel. nasa. gov/ career-resources/ project-management-and-systems-engineering-competency-model/.

DAU. 2013. *ENG Competency Model*, 12 June 2013 version. in Defense Acquisition University (DAU)/U.S. Department of Defense Database Online. Accessed on September 23, 2014. Available at https:/ / acc. dau. mil/ CommunityBrowser.aspx?id=657526&lang=en-US

Davidz, H.L. and J. Martin. 2011. "Defining a Strategy for Development of Systems Capability in the Workforce". *Systems Engineering.* 14(2): 141-143.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205*. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Ethical Behavior

If the competency(glossary) of the systems engineer is a matter of KSAA—knowledge, skills, abilities, and attitudes—then the word “attitudes” must have an ethical dimension. The ethical framework that guides the SE's actions insures that the SE ultimately does good and not harm. Ethical standards apply both to individuals and to organizations. This section discusses the moral foundations of ethics, and the elements of ethical conduct that are especially relevant to systems engineering.

Ethics and Morals in Systems Engineering

Like other people, systems engineers have morals: guiding personal thoughts and feelings about what is right and wrong. All of us also share, with other members of various communities to which we belong, ethics: standards that say what conduct is appropriate and what is not (Whitbeck 2007).

Morals are part of a person's character, the result of upbringing, culture, and other environmental influences. **Ethics** apply morals within the frame of a social system, which could be professional, business, academic, recreational, cultural, political, religious, or even familial. While a person’s moral code is usually considered immutable, one's ethics may need to account for new situations as one's profession or role in life changes. Tensions may exist between an engineer's responsibilities to society and those to the customer, the employer, or even the family, resulting in ethical dilemmas, and creating situations where morals come into play.

There is no shortage of discussion on ethics. Ethical codes are promulgated by professional and other organizations. **Professions** here refers to occupations that require learning and advanced knowledge and which safeguard or promote the well-being of others and of society as a whole.

Systems engineers have two ethical responsibilities over and above those of most other engineering professions:

- While engineers in general use their professional skills to address customer needs and desires, systems engineering (SE) helps *determine* those needs and desires in the course of defining and managing requirements. SEs have an obligation to ensure that problem or program definition is influenced solely by the interests of the customer or user, not by those of the systems engineer or his firm.
- Systems engineers typically integrate and oversee the work of others whose expertise differs from their own. This makes the obligation to widen one’s understanding and to seek competent advice from other professionals more acute in SE than in other disciplines.

Caroline Whitbeck's *Ethics in Engineering Practice and Research* explains what ethical behavior means for engineering professionals. Like most books on ethics, this one starts by clarifying the differences between ethics and morals, which can seem somewhat obscure at times (Whitbeck 2007).

A sampling of areas where ethics figure in the engineering of modern systems are described below.

Data Confidentiality and Security, Surveillance, and Privacy

Privacy, confidentiality, and security in systems which touch Personally Identifiable Information (PII) have an ethical dimension for the systems engineers responsible for developing those systems.

Laws and Regulations

Systems are typically developed in societies, sometimes involving international communities, which have laws concerning contracts, intellectual property, freedom of information, and employment. The requirements and restraints of those laws govern the practice of the systems engineer, who must be aware of the laws and must consider their implications for the partnerships that system development entails.

Whether or not they are stated in the system requirements document or provided by the customer, laws and regulations do in fact impose system requirements. SEs are responsible for knowing and applying relevant laws and regulations. This means recognizing other people’s proprietary interests by safeguarding their intellectual property (trade secrets, copyrights, trademarks, and patents), and giving them credit for performing work and making innovations.

Cultural Issues

Since systems engineers develop and maintain products used by humans globally, it is important that they understand the historical and cultural aspects of their profession and the related context in which their products will be used. System engineers need to be aware of societal diversity and act without prejudice or discrimination.

Ethical Considerations in the Systems Engineering Method

Naturally, SE approaches to meeting customer needs must integrate SE ethics.

Codes of Ethics and Professional Conduct

Codes of ethics are promulgated by the IEEE (IEEE 2009), the National Society of Professional Engineers (NSPE) (NSPE 2007), the International Council on Systems Engineering (INCOSE 2006) and other engineering organizations.

The INCOSE Code of Ethics enunciates fundamental ethical principles like honesty, impartiality, integrity, keeping abreast of knowledge, striving to increase competence, and supporting educational and professional organizations. Based on these principles, the code identifies the systems engineer's fundamental duties to society and the public, and the rules of practice that systems engineers should follow to fulfill those duties.

According to the INCOSE Code of Ethics, it is the systems engineer's duty to

- guard the public interest and protect the environment, safety, and welfare of those affected by engineering activities and technological artifacts;
- accept responsibility for one's actions and engineering results, including being open to ethical scrutiny and assessment;
- proactively mitigate unsafe practice;
- manage risk using knowledge gained by applying a whole-system viewpoint and understanding of systemic interfaces; and
- promote the understanding, implementation, and acceptance of prudent SE measures.

Enforcing Ethics

Many organizations enforce ethics internally by means of ethics policies. These polices typically include rules such as the following:

- There shall be no exchange of favors between anyone in the organization and entities with which it does business, such as suppliers, customers, or regulatory agencies.
- Product information, for example, test data, shall be reported accurately and completely to the contracting agency.
- There shall be no conflict of interest between the organization and entities with which it does business.

Favors can consist of providing money, reimbursement of travel or entertainment expenses, other items of equivalent value, or inappropriate job offers. Conflict of interest can arise when the personal or professional financial interests or organizational ties of an engineer are potentially at odds with the best interests of the customer or the engineer's employer. Since conflict of interest and other ethical transgressions can be hard to define, care must be taken to design ethics policies that are observable and enforceable. Internal audit functions or external regulatory agencies may enforce ethical rules at the individual, team, organizational, or enterprise level. Punishment for violating ethics policies can include termination and other disciplinary actions.

Unlike self-employed physicians who may choose to not do something specific, many systems engineers are individuals employed by organizations. Depending on the organizational context, an issue in conflict with the company might result in giving up the job. This may result in additional ethical considerations.

Responsibility to Society

Engineers who create products and services for use in society have an obligation to serve the public good. Additionally, the IEEE Code of Ethics states that engineers have an obligation to foster the professional development and ethical integrity of colleagues (IEEE 2015). Because of the criticality and scope of many systems, systems engineers, operating in teams within projects and on behalf of the public in delivery of products, have special responsibility. Poorly designed systems or services can have calamitous effects on society. The INCOSE Code of Ethics asserts the responsibility of systems engineers to "guard the public interest and protect the environment, safety, and welfare of those affected by engineering activities and technological artifacts" (INCOSE 2006).

References

Works Cited

IEEE. 2009. *IEEE Code of Ethics*. in IEEE [database online]. Accessed September 7, 2012. Available: http://www.ieee.org/ethics.

IEEE. 2015. *IEEE Code of Ethics*. Accessed April 6, 2015. Available: http://www.ieee.org/about/corporate/governance/p7-8.html.

INCOSE. 2006. *INCOSE Code of Ethics*. in International Council on Systems Engineering [database online]. Accessed April 13, 2015. Available: http://www.incose.org/about/leadershiporganization/codeofethics.

NSPE. 2007. *NSPE Code of Ethics for Engineers.* Alexandria, VA, USA: National Society of Professional Engineers. NSPE publication #1102. Accessed on September 15, 2011. Available: http://www.nspe.org/resources/ethics.

Whitbeck, C. 2007. *Ethics in Engineering Practice and Research*. New York, NY, USA: Cambridge University Press.

Primary References

Whitbeck, C. 2007. *Ethics in Engineering Practice and Research.* New York, NY, USA: Cambridge University Press.

Additional References

National Institute for Ethics in Engineering. "National Institute for Engineering Ethics." In the Murdough Center for for Engineering Professionalism. Hosted by Texas Technical University. Accessed on September 15, 2011. Available at http://www.murdough.ttu.edu/pd.cfm?pt=NIEE.

OnlineEthics.org. "Online Ethics Center (OEC)." Accessed September 8, 2011. Available at: http:/ / www. onlineethics.org/.

Martin, M. and R. Schinzinger. 2004. *Ethics in Engineering,* 4th ed. New York, NY, USA: McGraw-Hill.

Penn State. "Ethics: Books." In Penn State College of Engineering. Accessed September 8, 2011. Available at: http:/ /www.engr.psu.edu/ethics/books.asp.

Smith, J.H. (ed). 2008. *Engineering Ethics – Concepts, Viewpoint, Cases and Codes.* National Institute for Engineering Ethics, Texas Technical University.

< Previous Article | Parent Article | Next Article (Part 6)>

SEBoK v. 1.9.1, released 16 October 2018

Part 6: Related Disciplines

Related Disciplines

Part 6 of the Guide to the SE Body of Knowledge (SEBoK) presents knowledge that would be useful to systems engineers as they interact with these other fields and experts in those fields.

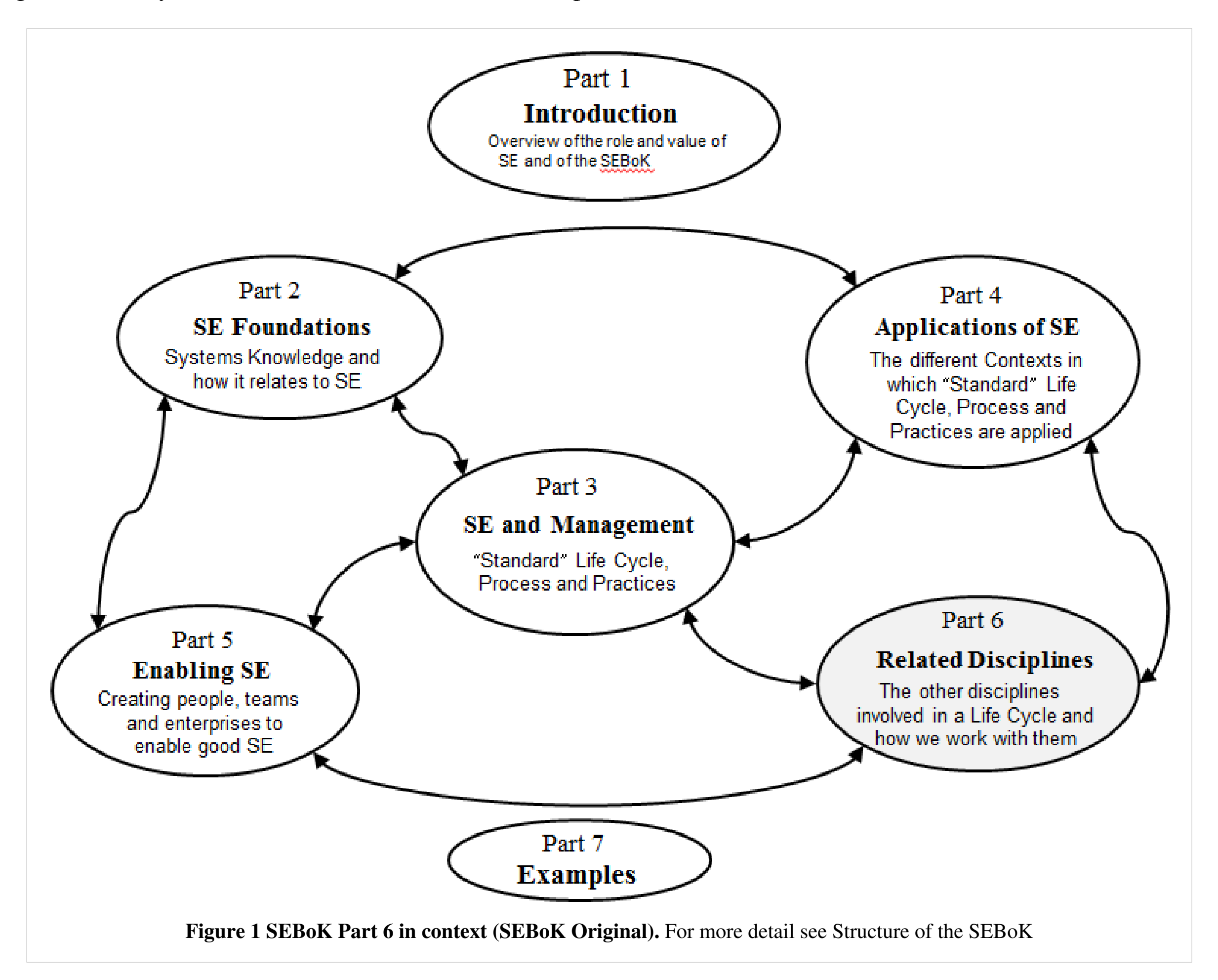


Figure 1 SEBoK Part 6 in context (SEBoK Original). For more detail see Structure of the SEBoK

Systems engineering (SE), as a discipline, intersects with other disciplines across the practice of engineering and across the enterprise. The knowledge areas (KAs) contained in this part, and the topics under them, are not meant to comprise additional bodies of knowledge but, rather, to give an overview with emphasis on what a systems engineer needs to know, accompanied by pointers to that knowledge.

Knowledge Areas in Part 6

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 6 contains the following KAs:

- Systems Engineering and Software Engineering
- Systems Engineering and Project Management
- Systems Engineering and Industrial Engineering
- Systems Engineering and Specialty Engineering

References

Works Cited

None.

Primary References

Bourque, P. and R.E. Fairley (eds.). 2014. *SWEBOK: Guide to the Software Engineering Body of Knowledge*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.Swebok.org.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense. February 19, 2010.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Systems Engineering and Software Engineering

Systems Engineering and Software Engineering

Software is prominent in most modern systems architectures and is often the primary means for integrating complex system components. Software engineering and systems engineering are not merely related disciplines; they are intimately intertwined. (See Systems Engineering and Other Disciplines.) Good systems engineering is a key factor in enabling good software engineering.

The SEBoK explicitly recognizes and embraces the intertwining between systems engineering and software engineering, as well as defining the relationship between the SEBoK and the Guide to the Software Engineering Body of Knowledge (SWEBOK) (Bourque, and Fairley, 2014).

This knowledge area describes the nature of software, provides an overview of the SWEBOK, describes the concepts that are shared by systems engineers and software engineers, and indicates the similarities and difference in how software engineers and systems engineers apply these concepts and use common terminology. It also describes the nature of the relationships between software engineering and systems engineering and describes some of the methods, models and tools used by software engineers.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Software Engineering in the Systems Engineering Life Cycle
- The Nature of Software
- An Overview of the SWEBOK Guide
- Key Points a Systems Engineer Needs to Know about Software Engineering
- Software Engineering Features - Models, Methods, Tools, Standards, and Metrics

Discussion

Software engineers, like systems engineers,

- engage in analysis and design, allocation of requirements, oversight of component development, component integration, verification and validation, life cycle sustainment, and system retirement.
- work with or as a component specialists (for example, user interface, database, computation, and communication specialists) who construct or otherwise obtain the needed software components.
- adapt existing components and incorporate components supplied by customers and affiliated organizations.

These commonalities would make it appear that software engineering is merely an application of systems engineering, but this is only a superficial appearance. The differences between the two disciplines arise from two fundamental issues:

1. Differences in educational backgrounds (traditional engineering disciplines for SE and the computing disciplines for SWE) and work experiences that result in different approaches to problem solving, and
2. Different ways of applying shared concepts based on the contrasting natures of the software medium and the physical media of traditional engineering.

Table 1 itemizes some of the shared concepts that are applied in different ways by systems engineers and software engineers. Each discipline has made contributions to the other. Table 1 indicates the methods and techniques developed by systems engineers adapted for use by software engineers and, conversely, those that have been adapted for use by systems engineers.

Table 1. Adaptation of Methods Across SE and SWE *

Systems Engineering Methods Adapted to Software Engineering	Software Engineering Methods Adapted to Systems Engineering
• Stakeholder Analysis	• Model-Driven Development
• Requirements Engineering	• UML-SysML
• Functional Decomposition	• Use Cases
• Design Constraints	• Object-Oriented Design
• Architectural Design	• Iterative Development
• Design Criteria	• Agile Methods
• Design Tradeoffs	• Continuous Integration
• Interface Specification	• Process Modeling
• Traceability	• Process Improvement
• Configuration Management	• Incremental V&V
• Systematic Verification And Validation	

The articles in this knowledge area give an overview of software and software engineering aimed at systems engineers. It also provides more details on the relationship between systems and software life cycles and some of the detailed tools used by software engineers. As systems become more dependent on software as a primary means of delivering stakeholder value the historical distinction between software and systems engineering may need to be challenged. This is a current area of joint discussion between the two communities which will affect the future knowledge in both SEBoK and SWEBoK.

References

Works Cited

Bourque, P. and Fairley, R.E. (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*). Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.Swebok.org.

Fairley, R.E. and Willshire M.J., 2011. *Teaching systems engineering to software engineering students*, *CSEET* 2011, Software Engineering Education and Training, p: 219-226, ISBN: 978-1-4577-0349-2.

Primary References

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK).* Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.Swebok.org.

Brooks, F. 1995. *The Mythical Man-Month,* Anniversary Edition. Boston, MA, USA: Addison Wesley Longman Inc.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, NJ, USA: John Wiley and Sons.

Additional References

Pressman, R. 2009. *Software Engineering: A Practitioner's Approach.* 7th Ed. New York, NY, USA: McGraw Hill.

Schneidewind, N. 2009. *Systems and Software Engineering with Applications*. New York, NY: Institute of Electrical and Electronics Engineers.

Sommerville, I. 2010. *Software Engineering.* 9th Ed. Boston, MA, USA: Addison Wesley.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Software Engineering in the Systems Engineering Life Cycle

This article describes how software engineering (SwE) life cycle processes integrate with the SE life cycle. A joint workshop organized by INCOSE, the Systems Engineering Research Center and the IEEE Computer Society was held to consider this relationship (Pyster et al, 2015). This workshop concluded that:

> *Software is fundamental to the performance, features, and value of most modern engineering systems. It is not merely part of the system, but often shapes the system architecture; drives much of its complexity and emergent behavior; strains its verification; and drives much of the cost and schedule of its development. Given how significant an impact software has on system development and given how complex modern systems are, one would expect the relationship between the disciplines of systems engineering (SE) and software engineering (SWE) to be well defined. However, the relationship is, in fact, not well understood or articulated.*

In this article we give some of the basic relationships between SwE and SE and discuss how these can be related to some of the SEBoK knowledge areas.

Systems Engineering and Software Engineering Life Cycles

The Guide to the Software Engineering Body of Knowledge (SWEBoK) (Bourque and Fairley 2014) describes the life cycle of a software product as:

- analysis and design,
- construction,
- testing,
- operation,
- maintenance, and eventually
- retirement or replacement.

This life cycle is common to most other mature engineering disciplines.

In Part 3 of the SEBoK, SE and Management, there is a discussion of SE life cycle models and life cycle processes. A Generic Life Cycle Model is described, and reproduced in Fig. 1 below. This is used to describe necessary stages in the life cycle of a typical engineered system.

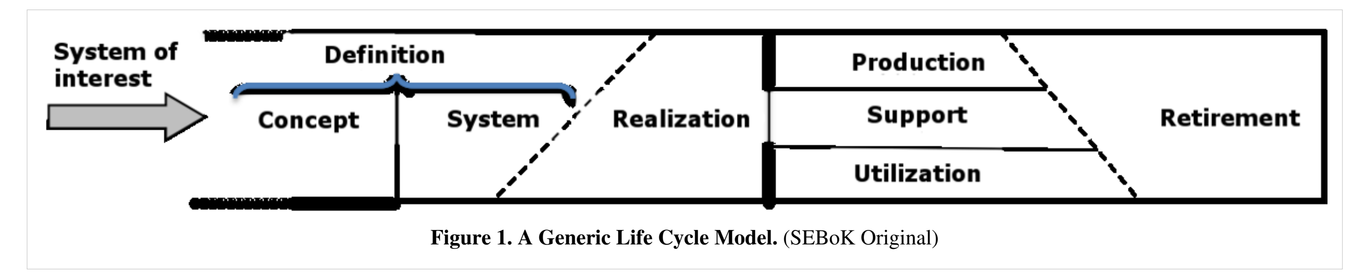


Figure 1. A Generic Life Cycle Model. (SEBoK Original)

Part 3 defines a collection of generic SE life cycle processes which define the activities and information needed across the SE life cycle. These processes include activities which contribute across the whole life cycle, with peaks of focused activity in certain stages, see Applying Life Cycle Processes for details.

The following sections provide a brief discussion of how SwE life cycle processes fit into SE life cycle process models. In practice, the details of this relationship are a key part of how a system life cycle is planned and delivered. The relationship will be shaped by the operating domain practice and solution type. Some examples of this are provided in the Implementation Examples.

Systems Engineering and Software Engineering Standards

The Systems Engineering life cycle processes described in Part 3, SE and Management, are largely based on those defined in the ISO/IEC/IEEE SE Life Cycle Processes 15288 Standard (2015).

The SWEBoK references the equivalent ISO/IECIEEE Software Engineering Life Cycle Processes 12207 Standard (2008), which defines a very similar set of processes for software systems. Figure 2 shows the relationship between the Enabling, Acquisition, Project and Technical Systems and Software processes in both 15288 and 12207 and the software specific processes of 12207. This alignment is from the last updates of both 12207 and 15288 in 2008. The SE processes have been further updated in 15288:2015, see Systems Engineering and Management for details. This change has not yet been applied to 12207. An update of 12207 is planned for 2016, in which the alignment to 15288 will be reviewed. See Alignment and Comparison of the Standards for more discussion of the relationships between the standards.

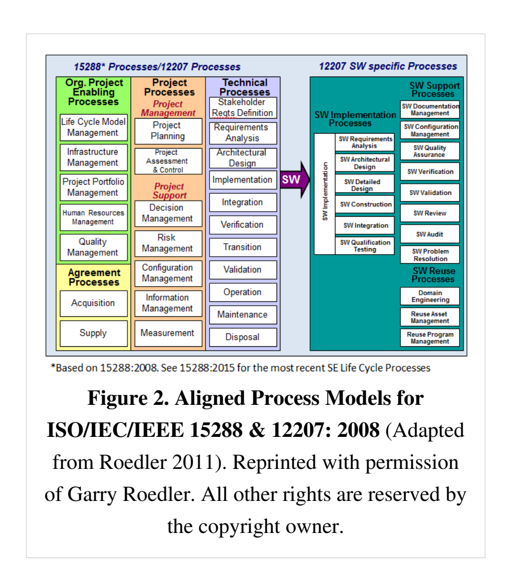


Figure 2. Aligned Process Models for ISO/IEC/IEEE 15288 & 12207: 2008 (Adapted from Roedler 2011). Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

Systems Engineering and Software Engineering Life Cycle Relationships

Pyster et al (2015) define two technical dimensions of engineered systems and of the engineering disciplines associated with them. The vertical dimensions of a system are those that modularize around technically focused engineering concerns involving specific elements of the system; the horizontal dimensions of a system involve cross-cutting concerns at the systems level. Examples of vertical concerns include quality attributes and performance effectiveness; and cost, schedule and risk of physical, organizational or human system elements associated with a particular technology domain. Examples of horizontal concerns include addressing evolving customer preferences that drive systems-level quality attributes, trade-off and optimization; resolving system architecture, decomposition and integration issues; implementing system development processes; and balancing system economics, cost, risk and schedule.

In complex systems projects, SE has a horizontal role while traditional engineering disciplines such as electrical, mechanical, and chemical engineering have vertical roles. To the extent that it is responsible for all aspects of the successful delivery of software related elements SwE can be considered as one of the vertical discipline. All of these traditional vertical disciplines will have some input to the horizontal dimension. However, the nature of software and its role in many complex systems makes SwE a critical discipline for many horizontal concerns. This is discussed further below.

The ISO/IEC/IEEE 12207 software engineering standard (2008) considers two situations:

- The life cycle of software product, containing minimal physical hardware, should use the software specific processes and a simple life cycle
- The life cycle of systems with a significant software content (sometimes called software intensive systems) should integrate the software processes into the SE life cycle

The second of these is the one relevant to the practice of SE.and requires a significant horizontal contribution from SwE

The relationship central to this is the way **SwE Implementation Processes** (see Fig 2) are used in the SE life cycle to support the implementation of software intensive system elements. This simple relationship must be seen in the context of the concurrency, iteration and recursion relationship between SE life cycle processes described in Applying Life Cycle Processes. This means that, in general, software requirements and architecture processes will be applied alongside system requirements and architecture processes; while software integration and test processes are applied alongside system integration, verification and validation processes. These interrelationships help with vertical software concerns, ensuring detailed software design and construction issues are considered at the system level. They also help with horizontal concerns, ensuring whole system issues are considered and are influenced by an understanding of software. See the Nature of Software for more details.

The ways these related processes work together will depend on the systems approach to solution synthesis used and how this influences the life cycle. If a top down approach is used, problem needs and system architecture will drive software implementation and realization. If a bottom up approach is used, the architecture of existing software will strongly influence both the system solution and the problem which can be considered. In Applying Life Cycle Processes a "middle-out" approach is described which combines these two ideas and is the most common way to develop systems. This approach needs a two-way relationship between SE and SwE technical processes.

The **SW Support Processes** may also play these vertical and horizontal roles. Part 3 contains knowledge areas on both System Deployment and Use which includes operation, maintenance and logistics; and Systems Engineering Management which covers the project processes shown in Figure 2. SwE support processes focus on the successful vertical deployment and use of software system elements and the management needed to achieve this. They also support their equivalent horizontal SE processes in contributing to the success of the whole system life cycle. The **Software Reuse Processes** have a particularly important role to play in deployment and use and Product and Service Life Management processes. The later considers Service Life Extension, Capability Updates, Upgrades, and

Modernization and system Disposal and Retirement. All of these horizontal software engineering activities rely on the associated SE activities having a sufficient understanding of the strengths and limitations of software and SwE, see Key Points a Systems Engineer Needs to Know about Software Engineering.

The Life Cycle Models knowledge area also defines how Vee and Iterative life cycle models provide a framework to tailor the generic life cycle and process definitions to different types of system development. Both models, with some modification, apply equally to the development of products and services containing software. Thus, the simple relationships between SE and SwE processes will form the basis for tailoring to suite project needs within a selected life cycle model.

Software and Systems Challenges

Pyster et al. (2015) define three classes of software intensive systems distinguished by the primary sources of novelty, functionality, complexity and risk in their conception, development, operation and evolution. These are briefly described below:

- **Physical Systems** operate on and generate matter or energy. While they often utilize computation and software technologies as components, those components are not dominant in the horizontal dimension of engineering. Rather, in such systems, they are defined as discrete system elements and viewed and handled as vertical concerns.
- **Computational Systems** include those in which computational behavior and, ipso facto, software are dominant at the systems level. The primary purpose of these systems is to operate on and produce data and information. While these systems always include physical and human elements, these are not the predominant challenges in system development, operation and evolution.
- **Cyber-Physical Systems** are a complex combination of computational and physical dimensions. Such systems are innovative, functionally complex and risky in both their cyber and physical dimensions. They pose major horizontal engineering challenges across the board. In cyber-physical systems, cyber and physical elements collaborate in complex ways to deliver expected system behavior.

Some of the challenges of physical and computational systems are well known and can be seen in many SE and SwE case studies. For example, physical system life cycles often make key decisions about the system architecture or hardware implementation which limit the subsequent development of software architecture and designs. This can lead to software which is inefficient and difficult or expensive to change. Problems which arise later in the life of such systems may be dealt with by changing software, or human, elements. This is sometimes done in a way which does not fully consider SwE design and testing practices. Similarly, computational systems may be dominated by the software architecture, without sufficient care taken to consider the best solutions for enabling hardware or people. In particular, operator interfaces, training and support may not be considered leading to the need for expensive organizational fixes once they are in use. Many computational systems in the past have been developed without a clear view of the user need they contribute to, or the other systems they must work with to do so. These and other related issues point to a need for system and software engineers with a better understanding of each other's disciplines. Pyster et al. considers how SE and SwE education might be better integrated to help achieve this aim.

Examples of cyber-physical systems increasingly abound – smart automobiles, power grids, robotic manufacturing systems, defense and international security systems, supply-chain systems, the so-called internet of things, etc. In these systems there is no clear distinction between software elements and the whole system solution. The use of software in these systems is central to the physical outcome and software is often the integrating element which brings physical elements and people together. These ideas are closely align with the Service System Engineering approach described in Part 4.

SEBoK Part 3 includes a Business and Mission Analysis process which is based on the equivalent process in the updated ISO/IEC/IEEE 15288 (2015). This process enables SE to be involved in the selection and bounding of the problem situation which forms the starting point for an engineered system life cycle. For cyber physical systems an

understanding of the nature of software is needed in the formulation of the problem, since this is often fundamentally driven by the use of software to create complex adaptive solution concepts. This close coupling of software, physical and human system elements across the system of interest continues throughout the system life cycle making it necessary to consider all three in most horizontal system level decision.

The life cycle of cyber physical systems cannot be easily partitioned into SE and SwE achieving their own outcomes, but working together on horizontal system issues. It will require a much more closely integrated approach, requiring systems and software engineers with a complementary set of competencies, and changes how the two disciplines are seen in both team and organizational structures. See Enabling Systems Engineering.

References

Works Cited

Pyster, A., Adcock, R., Ardis, M., Cloutier, R., Henry, D., Laird, L., Lawson, H. 'Bud'., Pennotti, M., Sullivan, K., Wade J. 2015. Exploring the Relationship between Systems Engineering and Software Engineering. 13th Conference on Systems Engineering Research (CSER), Procedia Computer Science, Volume 44, 2015, Pages 708-717

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IECIEEE. 2008. Systems and Software Engineering — Software Life Cycle Processes. Geneva, Switzerland: International Organization for Standards (ISO)/Institute of Electrical & Electronics Engineers (IEEE) Computer Society, ISO//IECIEEE 12207:2008(E).

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

Primary References

Pyster, A., Adcock, R., Ardis, M., Cloutier, R., Henry, D., Laird, L., Lawson, H. 'Bud'., Pennotti, M., Sullivan, K., Wade J. 2015. Exploring the Relationship between Systems Engineering and Software Engineering. 13th Conference on Systems Engineering Research (CSER), Procedia Computer Science, Volume 44, 2015, Pages 708-717

Additional References

Roedler, G. 2010. *An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes.* Asian Pacific Council on Systems Engineering (APCOSE) Conference.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

The Nature of Software

The nature of the software medium has many consequences for systems engineering (SE) of software-intensive systems. Fred Brooks has famously observed that four properties of software, taken together, differentiate it from other kinds of engineering artifacts (Brooks 1995). These four properties are

1. complexity,
2. conformity,
3. changeability
4. invisibility.

Brooks states:

> *Software entities are more complex for their size than perhaps any other human construct because no two parts are alike (at least above the statement level). If they are, we make the two similar parts into a subroutine — open or closed. In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound.* (Brooks 1995, p 82)

Complexity

The complexity of software arises from the large number of unique interacting parts in a software system. The parts are unique because they are encapsulated as functions, subroutines, or objects, and invoked as needed rather than being replicated. Software parts have several different kinds of interactions, including serial and concurrent invocations, state transitions, data couplings, and interfaces to databases and external systems.

Depiction of a software entity often requires several different design representations to portray the numerous static structures, dynamic couplings, and modes of interaction that exist in computer software. Complexity within the parts, and in the connections among parts requires that changes undergo substantial design rigor and regression testing. Software provides functionality for components that are embedded, distributed and data centric. Software can implement simple control aoop s well as complex algorithms and heuristics.

Complexity can hide defects that may not be discovered easily, thus requiring significant additional and unplanned rework.

Conformity

Software, unlike a physical product, has no underlying natural principles which it must conform to, such as Newton's laws of motion. However, software must conform to exacting specifications in the representation of each of its parts, in the interfaces to other internal parts, and in the connections to the environment in which it operates. A missing semicolon or other syntactic error can be detected by a compiler. But, a defect in the program logic or a timing error may be difficult to detect when encountered during operation.on.

Unlike software, tolerance among the interfaces of physical entities is the foundation of manufacturing and assembly. No two physical parts that are joined together have, or are required to have, exact matches. There are no corresponding tolerances in the interfaces among software entities or between software entities and their environments. There are no interface specifications for software stating that a parameter can be *an integer plus or minus 2%.* Interfaces among software parts must agree exactly in numbers, types of parameters and kinds of couplings.

Lack of conformity can cause problems when an existing software component cannot be reused as planned because it does not conform to the needs of the product under development. Lack of conformity might not be discovered until late in a project, thus necessitating the development and integration of an acceptable component to replace the one that cannot be reused. This requires an unplanned allocation of resources (usually) and can delay project completion.

Changeability

Software coordinates the operation of physical components and provides most of the functionality in software-intensive systems. Because software is the most malleable (easily changed) element in a software-intensive system, it is the most frequently changed element. This is particularly true during the late stages of a development project and during system sustainment. However, this does not mean that software is easy to change. Complexity and the need for conformity can make changing software an extremely difficult task. Changing one part of a software system often results in undesired side effects in other parts of the system, requiring more changes before the software can operate at maximum efficiency.

Invisibility

Software is said to be invisible because it has no physical properties. While the effects of executing software on a digital computer are observable, software itself cannot be seen, tasted, smelled, touched, or heard. Software is an intangible entity because our five human senses are incapable of directly sensing it.

Work products such as requirements specifications, design documents, source code and object code are representations of software, but they are not the software. At the most elemental level, software resides in the magnetization and current flow in an enormous number of electronic elements within a digital device. Because software has no physical presence, software engineers must use different representations at different levels of abstraction in an attempt to visualize the inherently invisible entity.

Uniqueness

One other point about the nature of software that Brooks alludes to, but does not explicitly call out is the uniqueness of software. Software and software projects are unique for the following reasons:

- Software has no physical properties;
- Software is the product of intellect-intensive teamwork;
- Productivity of software developers varies more widely than the productivity of other engineering disciplines;
- Estimation and planning for software projects is characterized by a high degree of uncertainty, which can be at best partially mitigated by best practices;
- Risk management for software projects is predominantly process-oriented;
- Software alone is useless, as it is always a part of a larger system; and
- Software is the most frequently changed element of software intensive systems.

References

Works Cited

Brooks, F. 1995. *The Mythical Man-Month,* Anniversary Edition. Boston, MA, USA: Addison Wesley Longman Inc.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, New Jersey: John Wiley and Sons.

Primary References

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK).* Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.Swebok.org.

Brooks, F. 1995. *The Mythical Man-Month,* Anniversary Edition. Boston, MA, USA: Addison Wesley Longman Inc.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, New Jersey: John Wiley and Sons.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

An Overview of the SWEBOK Guide

Systems engineers are fortunate that the software community has developed its own body of knowledge. The introduction to Version 3 of the *Guide to the Software Engineering Body of Knowledge* states:

> *The purpose of the Guide is to describe the portion of the Body of Knowledge that is generally accepted, to organize that portion, and to provide topical access to it.* (Bourque and Fairley 2014)

SWEBOK Guide Version 3

The purposes of SWEBOK V3 are as follows:

- to characterize the contents of the software engineering discipline;
- to promote a consistent view of software engineering worldwide;
- to clarify the place of, and set the boundary of, software engineering with respect to other disciplines;
- to provide a foundation for training materials and curriculum development; and
- to provide a basis for certification and licensing of software engineers.

SWEBOK V3 contains 15 knowledge areas (KAs). Each KA includes an introduction, a descriptive breakdown of topics and sub-topics, recommended references, references for further reading, and a matrix matching reference material with each topic. An appendix provides a list of standards most relevant to each KA. An overview of the individual KAs presented in the guide is provided in the next two sections.

Knowledge Areas Characterizing the Practice of Software Engineering

Software Requirements

The Software Requirements KA is concerned with the elicitation, negotiation, analysis, specification, and validation of software requirements. It is widely acknowledged within the software industry that software engineering projects are critically vulnerable when these activities are performed poorly. Software requirements express the needs and constraints placed on a software product that contribute to the solution of some real-world problems.

Software Design

Design is defined as both *the process of defining the architecture, components, interfaces, and other characteristics of a system or component* and *the result of [that] process* (IEEE 1991). The Software Design KA covers the design process and the resulting product. The software design process is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software’s internal structure and its behavior that will serve as the basis for its construction. A software design (the result) must describe the software architecture -- that is, how software is decomposed and organized into components and the interfaces between those components. It must also describes the components at a level of detail that enables their construction.

Software Construction

Software construction refers to the detailed creation of working software through a combination of detailed design, coding, unit testing, integration testing, debugging, and verification. The Software Construction KA includes topics related to the development of software programs that will satisfy their requirements and design constraints. This KA covers software construction fundamentals; managing software construction; construction technologies; practical considerations; and software construction tools.

Software Testing

Testing is an activity performed to evaluate product quality and to improve it by identifying defects. Software testing involves dynamic verification of the behavior of a program against expected behavior on a finite set of test cases. These test cases are selected from the (usually very large) execution domain. The Software Testing KA includes the fundamentals of software testing; testing techniques; human-computer user interface testing and evaluation; test-related measures; and practical considerations.

Software Maintenance

Software maintenance involves enhancing existing capabilities, adapting software to operate in new and modified operating environments, and correcting defects. These categories are referred to as perfective, adaptive, and corrective software maintenance. The Software Maintenance KA includes fundamentals of software maintenance (nature of and need for maintenance, categories of maintenance, maintenance costs); key issues in software maintenance (technical issues, management issues, maintenance cost estimation, measurement of software maintenance); the maintenance process; software maintenance techniques (program comprehension, re-engineering, reverse engineering, refactoring, software retirement); disaster recovery techniques, and software maintenance tools.

Software Configuration Management

The configuration of a system is the functional and/or physical characteristics of hardware, firmware, software, or a combination of these. It can also be considered as a collection of specific versions of hardware, firmware, or software items combined according to specific build procedures to serve a particular purpose. Software configuration management (SCM) is thus the discipline of identifying the configuration of a system at distinct points in time for the purposes of systematically controlling changes to the configuration, as well as maintaining the integrity and traceability of the configuration throughout the software life cycle. The Software Configuration Management KA covers management of the SCM process; software configuration identification, control, status accounting, auditing; software release management and delivery; and software configuration management tools.

Software Engineering Management

Software engineering management involves planning, coordinating, measuring, reporting, and controlling a project or program to ensure that development and maintenance of the software is systematic, disciplined, and quantified. The Software Engineering Management KA covers initiation and scope definition (determining and negotiating requirements, feasibility analysis, and review and revision of requirements); software project planning (process planning, estimation of effort, cost, and schedule, resource allocation, risk analysis, planning for quality); software project enactment (measuring, reporting, and controlling; acquisition and supplier contract management); product acceptance; review and analysis of project performance; project closure; and software management tools.

Software Engineering Process

The Software Engineering KA is concerned with definition, implementation, assessment, measurement, management, and improvement of software life cycle processes. Topics covered include process implementation and change (process infrastructure, models for process implementation and change, and software process management); process definition (software life cycle models and processes, notations for process definition, process adaptation, and process automation); process assessment models and methods; measurement (process measurement, products measurement, measurement techniques, and quality of measurement results); and software process tools.

Software Engineering Models and Methods

The Software Engineering Models and Methods KA addresses methods that encompass multiple life cycle stages; methods specific to particular life cycle stages are covered by other KAs. Topics covered include modeling (principles and properties of software engineering models; syntax vs. semantics vs. invariants; preconditions, post-conditions, and invariants); types of models (information, structural, and behavioral models); analysis (analyzing for correctness, completeness, consistency, quality and interactions; traceability; and tradeoff analysis); and software development methods (heuristic methods, formal methods, prototyping methods, and agile methods).

Software Quality

Software quality is a pervasive software life cycle concern that is addressed in many of the SWEBOK V3 KAs. In addition, the Software Quality KA includes fundamentals of software quality (software engineering cultures, software quality characteristics, the value and cost of software quality, and software quality improvement); software quality management processes (software quality assurance, verification and validation, reviews and audits); and practical considerations (defect characterization, software quality measurement, and software quality tools).

Software Engineering Professional Practice

Software engineering professional practice is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner. The Software Engineering Professional Practice KA covers professionalism (professional conduct, professional societies, software engineering standards, employment contracts, and legal issues); codes of ethics; group dynamics (working in teams, cognitive problem complexity, interacting with stakeholders, dealing with uncertainty and ambiguity, dealing with multicultural environments); and communication skills.

Knowledge Areas Characterizing the Educational Requirements of Software Engineering

Software Engineering Economics

The Software Engineering Economics KA is concerned with making decisions within the business context to align technical decisions with the business goals of an organization. Topics covered include fundamentals of software engineering economics (proposals, cash flow, the time-value of money, planning horizons, inflation, depreciation, replacement and retirement decisions); not for-profit decision-making (cost-benefit analysis, optimization analysis); estimation, economic risk and uncertainty (estimation techniques, decisions under risk and uncertainty); and multiple attribute decision making (value and measurement scales, compensatory and non-compensatory techniques).

Computing Foundations

The Computing Foundations KA covers fundamental topics that provide the computing background necessary for the practice of software engineering. Topics covered include problem solving techniques, abstraction, algorithms and complexity, programming fundamentals, the basics of parallel and distributed computing, computer organization, operating systems, and network communication.

Mathematical Foundations

The Mathematical Foundations KA covers fundamental topics that provide the mathematical background necessary for the practice of software engineering. Topics covered include sets, relations, and functions; basic propositional and predicate logic; proof techniques; graphs and trees; discrete probability; grammars and finite state machines; and number theory.

Engineering Foundations

The Engineering Foundations KA covers fundamental topics that provide the engineering background necessary for the practice of software engineering. Topics covered include empirical methods and experimental techniques; statistical analysis; measurements and metrics; engineering design; simulation and modeling; and root cause analysis.

Related Disciplines

SWEBOK V3 also discusses related disciplines. The related disciplines are those that share a boundary, and often a common intersection, with software engineering. SWEBOK V3 does not characterize the knowledge of the related disciplines but, rather, indicates how those disciplines interact with the software engineering discipline. The related disciplines include

- Computer Engineering
- Computer Science
- General Management
- Mathematics
- Project Management
- Quality Management
- Systems Engineering

References

Works Cited

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Primary References

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Key Points a Systems Engineer Needs to Know about Software Engineering

The field of Software Engineering (glossary) is extensive and specialized. Its importance to modern systems makes it necessary for systems engineers to be knowledgeable about software engineering and its relationship to systems engineering.

Key Concepts a Systems Engineer Needs to Know about Software Engineering

The following items are significant aspects that systems engineers need to know about software and software engineering. Most are documented in (Fairley and Willshire 2011):

1. **For the time, effort, and expense devoted to developing it, software is more complex than most other system components** - Software complexity arises because few elements in a software program (even down to the statement level) are identical as well as because of the large number of possible decision paths found even in small programs, with the number of decision paths through a large program often being astronomical. There are several detailed references on software complexity. The SWEBOK (Bourque and Fairley 2014) discusses minimizing complexity as part of software construction fundamentals. Zuse (1991) has a highly cited article on software complexity measures and methods. Chapters 2 and 3 of the SWEBOK also have further references.
2. **Software testing and reviews are sampling processes** - In all but the simplest cases, exhaustive testing of software is impossible because of the large number of decision paths through most programs. Also, the combined values of the input variables selected from a wide combinatorial range may reveal defects that other combinations of the variables would not detect. Software test cases and test scenarios are chosen in an attempt to gain confidence that the testing samples are representative of the ways the software will be used in practice. Structured reviews of software are an effective mechanism for finding defects, but the significant effort required limits exhaustive reviewing. Criteria must be established to determine which components (or sub-components) should be reviewed. Although there are similar concerns about exhaustive testing and reviewing of physical products, the complexity of software makes software testing, reviews, and the resulting assurance provided, more challenging. Other points include:
    1. All software testing approaches and techniques are heuristic. Hence, there is no universal "best" approach, practice, or technique for testing, since these must be selected based on the software context.
    2. Exhaustive testing is not possible.
    3. Errors in software tend to cluster within the software structures; therefore, any one specific approach or a random approach to testing is not advised.
    4. Pesticide paradox exists. As a result, running the same test over and over on the same software-system provides no new information.
    5. Testing can reveal the presence of defects but cannot guarantee that there will be no errors, except under the specific conditions of a given test.
    6. Testing, including verification and validation (V&V), must be performed early and continually throughout the lifecycle (end to end.
    7. Even after extensive testing and V&V, errors are likely to remain after long term use of the software.
    8. Chapter 4 of the SWEBOK discusses software testing and provides a bibliography.

3. **Software often provides the interfaces that interconnect other system components** - Software is often referred to as the *glue* that holds a system together because the interfaces among components, as well as the interfaces to the environment and other systems, are often provided by digital sensors and controllers that operate via software. Because software interfaces are behavioral rather than physical, the interactions that occur among software components often exhibit emergent behaviors that cannot always be predicted in advance. In addition to component interfaces, software usually provides the computational and decision algorithms needed to generate command and control signals. The SWEBOK has multiple discussions of interfaces: Chapter 2 on Software Design is a good starting point and includes a bibliography.
4. **Every software product is unique** - The goal of manufacturing physical products is to produce replicated copies that are as nearly identical as much as possible, given the constraints of material sciences and manufacturing tools and techniques. Because replication of existing software is a trivial process (as compared to manufacturing of physical products), the goal of software development is to produce one perfect copy (or as nearly perfect as can be achieved given the constraints on schedule, budget, resources, and technology). Much of software development involves altering existing software. The resulting product, whether new or modified, is uniquely different from all other software products known to the software developers. Chapter 3 of the SWEBOK provides discussion of software reuse and several references.
5. **In many cases, requirements allocated to software must be renegotiated and reprioritized** - Software engineers often see more efficient and effective ways to restate and prioritize requirements allocated to software. Sometimes, the renegotiated requirements have system-wide impacts that must be taken into account. One or more senior software engineers should be, and often are, involved in analysis of system-level requirements. This topic is addressed in the SWEBOK in Chapter 1, with topics on the iterative nature of software and change management.
6. **Software requirements are prone to frequent change** - Software is the most frequently changed component in complex systems, especially late in the development process and during system sustainment. This is due to the fact that software is perceived to be the most easily changed component of a complex system. This is not to imply that changes to software requirements, and the resulting changes to the impacted software, can be easily done without undesired side effects. Careful software configuration management is necessary, as discussed in Chapter 6 of the SWEBOK, which includes extensive references.
7. **Small changes to software can have large negative effects** (A corollary to frequently changing software requirements: *There are no small software changes*) - In several well-known cases, modifying a few lines of code in very large systems that incorporated software negatively impacted the safety, security, and/or reliability of those systems. Applying techniques such as traceability, impact analysis, object-oriented software development, and regression testing reduces undesired side effects of changes to software code. These approaches limit but do not eliminate this problem.
8. **Some quality attributes for software are subjectively evaluated** - Software typically provides the interfaces to systems that have human users and operators. The intended users and operators of these systems often subjectively evaluate quality attributes, such as ease of use, adaptability, robustness, and integrity. These quality attributes determine the acceptance of a system by its intended users and operators. In some cases, systems have been rejected because they were not judged to be suitable for use by the intended users in the intended environment, even though those systems satisfied their technical requirements. Chapter 10 of the SWEBOK provides an overview of software quality, with references.
9. **The term *prototyping* has different connotations for systems engineers and software engineers** - For a systems engineer, a prototype is typically the first functioning version of a hardware. For software engineers, software prototyping is primarily used for two purposes: (1) as a mechanism to elicit user requirements by iteratively evolving mock-ups of user interfaces, and (2) as an experimental implementation of some limited element of a proposed system to explore and evaluate alternative algorithms. Chapter 1 of the SWEBOK discusses this and provides excellent references.

10. **Cyber security is a present and growing concern for systems that incorporate software** - In addition to the traditional specialty disciplines of safety, reliability, and maintainability, systems engineering teams increasingly include security specialists at both the software level and the systems level in an attempt to cope with the cyber attacks that may be encountered by systems that incorporate software. Additional information about security engineering can be found in the Systems Engineering and Specialty Engineering KA.
11. **Software growth requires spare capacity** - Moore's Law no longer fully comes to the rescue (Moore, 1965). As systems adapt to changing circumstances, the modifications can most easily be performed and upgraded in the software, requiring additional computer execution cycles and memory capacity (Belady and Lehman 1979). For several decades, this growth was accommodated by Moore's Law, but recent limits that have occurred as a result of heat dissipation have influenced manufacturers to promote potential computing power growth by slowing down the processors and putting more of them on a chip. This requires software developers to revise their programs to perform more in parallel, which is often an extremely difficult problem (Patterson 2010). This problem is exacerbated by the growth in mobile computing and limited battery power.
12. **Several Pareto 80-20 distributions apply to software** - These refers to the 80% of the avoidable rework that comes from 20% of the defects, that 80% of the defects come from 20% of the modules, and 90% of the downtime comes from at most 10% of the defects (Boehm and Basili 2001). These, along with recent data indicating that 80% of the testing business value comes from 20% of the test cases (Bullock 2000), indicate that much more cost-effective software development and testing can come from determining which 20% need the most attention.
13. **Software estimates are often inaccurate** - There are several reasons software estimates are frequently inaccurate. Some of these reasons are the same as the reasons systems engineering estimates are often inaccurate: unrealistic assumptions, vague and changing requirements, and failure to update estimates as conditions change. In addition, software estimates are often inaccurate because productivity and quality are highly variable among seemingly similar software engineers. Knowing the performance characteristics of the individuals who will be involved in a software project can greatly increase the accuracy of a software estimate. Another factor is the cohesion of the software development team. Working with a team that has worked together before and knowing their collective performance characteristics can also increase the accuracy of a software estimate. Conversely, preparing an estimate for unknown teams and their members can result in a very low degree of accuracy. Chapter 7 of the SWEBOK [1] briefly discusses this further. Kitchenam (1997) discusses the organizational context of uncertainty in estimates. Lederer and Prasad (1995) also identify organizational and management issues that increase uncertainty; additionally, a recent dissertation from Sweden by Magazinus (2012) shows that the issues persist.
14. **Most software projects are conducted iteratively** - "Iterative development" has a different connotation for systems engineers and software engineers. A fundamental aspect of iterative software development is that each iteration of a software development cycle adds features and capabilities to produce a next working version of partially completed software. In addition, each iteration cycle for software development may occur on a daily or weekly basis, while (depending on the scale and complexity of the system) the nature of physical system components typically involves iterative cycles of longer durations. Classic articles on this include (Royce 1970) and (Boehm 1988), among others. Larman and Basili (2003) provide a history of iterative development, and the SWEBOK discusses this in life cycle processes in Chapter 8.
15. **Teamwork within software projects is closely coordinate** - The nature of software and its development requires close coordination of work activities that are predominately intellectual in nature. Certainly other engineers engage in intellectual problem solving, but the collective and ongoing daily problem solving required of a software team requires a level of communication and coordination among software developers that is of a different more elevated type. Highsmith (2000) gives a good overview.
16. **Agile development processes are increasingly used to develop software** - Agile development of software is a widely used and growing approach to developing software. Agile teams are typically small and closely

coordinated, for the reasons cited above. Multiple agile teams may be used on large software projects, although this is highly risky without an integrating architecture (Elssamadisy and Schalliol 2002). Agile development proceeds iteratively in cycles that produce incremental versions of software, with cycle durations that vary from one day to one month, although shorter durations are more common. Among the many factors that distinguish agile development is the tendency to evolve the detailed requirements iteratively. Most agile approaches do not produce an explicit design document. Martin (2003) gives a highly cited overview.

17. **Verification and validation (V&V) of software should preferably proceed incrementally and iteratively** - Iterative development of working product increments allows incremental verification, which ensures that the partial software product satisfies the technical requirements for that incremental version; additionally, it allows for the incremental validation (ISO/IEC/IEEE 24765) of the partial product to make certain that it satisfies its intended use, by its intended users, in its intended environment. Incremental verification and validation of working software allows early detection and correction of encountered problems. Waiting to perform integration, verification, and validation of complex system until later life cycle stages, when these activities are on the critical path to product release, can result in increased cost and schedule impacts. Typically, schedules have minimal slack time during later stages in projects. However, with iterative V&V, software configuration management processes and associated traceability aspects may become complex and require special care to avoid further problems. Chapter 4 of the SWEBOK discusses software testing, and provides numerous references, including standards. Much has been written on the subject; a representative article is (Wallace and Fujii 1989).
18. **Performance trade-offs are different for software than systems** - Systems engineers use “performance” to denote the entire operational envelope of a system; whereas, software engineers use “performance” to mean response time and the throughput of software. Consequentially, systems engineers have a larger design space in which to conduct trade studies. In software, performance is typically enhanced by reducing other attributes, such as security or ease of modification. Conversely, enhancing attributes such as security and ease of modification typically impacts performance of software (response time and throughput) in a negative manner.
19. **Risk management for software projects differs in kind from risk management for projects that develop physical artifacts** - Risk management for development of hardware components is often concerned with issues such as supply chain management, material science, and manufacturability. Software and hardware share some similar risk factors: uncertainty in requirements, schedule constraints, infrastructure support, and resource availability. In addition, risk management in software engineering often focuses on issues that result from communication problems and coordination difficulties within software development teams, across software development teams, and between software developers and other project members (e.g., hardware developers, technical writers, and those who perform independent verification and validation). See (Boehm 1991) for a foundational article on the matter.
20. **Software metrics include product measures and process measures** - The metrics used to measure and report progress of software projects include product measures and process (ISO/IEC/IEEE 24765) measures. Product measures include the amount of software developed (progress), defects discovered (quality), avoidable rework (defect correction), and budgeted resources used (technical budget, memory and execution cycles consumed, etc.). Process measures include: the amount of effort expended (because of the people-intensive nature of software development), productivity (software produced per unit of effort expended), production rate (software produced per unit time), milestones achieved and missed (schedule progress), and budgeted resources used (financial budget). Software metrics are often measured on each (or, periodically, some) of the iterations of a development project that produces a next working version of the software. Chapter 8 and Chapter 7 of the SWEBOK address this.
21. **Progress on software projects is sometimes inadequately tracked** - In some cases, progress on software projects is not adequately tracked because relevant metrics are not collected and analyzed. A fundamental problem is that accurate tracking of a software project depends on knowing how much software has been developed that is suitable for delivery into the larger system or into a user environment. Evidence of progress,

nthe form of working software, is one of the primary advantages of the iterative development of working software increments.

References

Works Cited

Belady, L. and M. Lehman. 1979. "Characteristics of Large Systems." In P. Wegner (ed.), *Research Directions in Software Technology*. Cambridge, MA, USA: MIT Press.

Boehm, B. 1991. "Software Risk Management: Principles and Practices." *IEEE Software*. 8(1):32-41.

Boehm, B. and V. Basili. 2001. "Software defect reduction Top 10 List." *Computer*. 34(1):135-137.

Brooks, F. 1995. *The Mythical Man-Month, Anniversary Edition*. Boston, MA, USA: Addison Wesley Longman Inc.

Bullock, J. 2000. "Calculating the Value of Testing." *Software Testing and Quality Engineering*, May-June, 56-62.

DeMarco, T. and T. Lister. 1987. *Peopleware: Predictive Projects and Teams*. New York, NY, USA: Dorset House.

Elssamadisy, A. and G. Schalliol. 2002. "Recognizing and Responding to 'Bad Smells' in Extreme Programming." *Proceedings, ICSE 2002*, ACM-IEEE, 617-622.

Fairley, R.E. and M.J. Willshire. 2011. "Teaching software engineering to undergraduate systems engineering students." *Proceedings of the 2011 American Society for Engineering Education (ASEE) Annual Conference and Exposition*. 26-29 June 2011. Vancouver, BC, Canada.

Kitchenham, B. 1997. "Estimates, Uncertainty, and Risk." *IEEE Software*. 14(3): 69-74.

Larman, C. and V.R. Basili. 2003. "Iterative and incremental developments: a brief history." *Computer*. 36(6): 47-56.

Lederer, A.L. and J. Prasad. 1995. "Causes of inaccurate software development cost estimates." *Journal of Systems and Software*. 31(2):125-134.

Magazinius, A. 2012. *Exploring Software Cost Estimation Inaccuracy*. Doctoral Dissertation. Chalmers University of Technology. Goteborg, SE.

Martin, R.C. 2002. *Agile Software Development: Principles, Patterns and Practices*. Upper Saddle River, NJ, USA:Prentice Hall.

Moore, G.E. 1965. "Cramming more components onto integrated circuits," *Electronics Magazine*, April 19, 4.

Patterson, D. 2010. "The Trouble With Multicore." *IEEE Spectrum*, July, 28-32, 52-53.

Royce, W.W. 1970. "Managing the development of large software systems." *Proceedings of IEEE WESCON*. August, 1970.

Wallace, D.R. and R.U. Fujii. 1989. "Software verification and validation: an overview." *IEEE Software*. 6(3):10-17.

Zuse, Horst. 1991. *Software Complexity: measures and methods*. Hawthorne, NJ, USA: Walter de Gruyter and Co.

Primary References

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.Swebok.org.

Brooks, Fred 1995. The Mythical Man-Month, Anniversary Edition. Reading, Massachusetts: Addison Wesley.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, NJ, USA: John Wiley & Sons.

PMI. 2013A. *A Guide to the Project Management Body of Knowledge|(PMBOK® Guide)*. 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

PMI 2013B. *Software Extension to the PMBOK® Guide*, Fifth Edition, Newtown Square, PA, USA: Project Management Institute (PMI) and Los Alamitos, CA, USA: IEEE Computer Society.

Pyster, A., M. Ardis, D. Frailey, D. Olwell, A. Squires. 2010. "Global workforce development projects in software engineering." *Crosstalk - The Journal of Defense Software Engineering*, Nov/Dec, 36-41. Available at: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA535633 Accessed 02 Dec 2015.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Software Engineering Features - Models, Methods, Tools, Standards, and Metrics

In recent decades software has become ubiquitous. Almost all modern engineered systems include significant software subsystems; this includes systems in the transportation, finance, education, healthcare, legal, military, and business sectors. Along with the increase in software utility, capability, cost, and size there has been a corresponding growth in methods, models, tools, metrics and standards, which support software engineering.

Chapter 10 of the SWEBOK discusses modeling principles and types, and the methods and tools that are used to develop, analyze, implement, and verify the models. The other SWEBOK chapters on the software development phases (e.g., Software Design) discuss methods and tools specific to the phase. Table 1 identifies software engineering features for different life-cycle phases. The table is not meant to be complete; it simply provide examples. In Part 2 of the SEBoK there is a discussion of models and the following is one of the definitions offered: “an abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system” (Dori 2002).

For the purposes of Table 1 we extend the definition of a model to some aspect of the software system or it development. So as an example, we list “Project Plan” as a model in the Software Management area. The idea is that the Project Plan provides a model of how the project is going to be carried out: the project team organization, the process to be used, the work to be done, the project schedule, and the resources needed.

Table 1: SWE Features (SEBoK Original)

Life-Cycle Activity	Models	Methods & Tools	Standards
Software Management	• Life-Cycle Process Model • Work Breakdown Structure • Constructive Cost Model (COCOMO) • Project Plan • Configuration Management (CM) Plan • Risk Management Plan	• Effort, Schedule and Cost Estimation • Risk Analysis • Data Collection • Project Tracking • CM Management • Iterative/Incremental Development • Agile Development	• [IEEE 828] • [IEEE 1058] • [IEEE 1540] • [IEEE 12207]
Software Requirements	• Functional Model • User Class Model • Data Flow Diagram • Object Model • Formal Model • User Stories	• Requirements Elicitation • Prototyping • Structural Analysis • Data-Oriented Analysis • Object-Oriented Analysis • Object Modeling Language (OML) • Formal Methods • Requirements Specification • Requirements Inspection	• [IEEE 830] • [IEEE 1012] • [IEEE 12207]

Software Design	• Architectural Model • Structure Diagram • Object Diagram • Class Specification • Data Model	• Structured Design • Object-Oriented Design • OML • Modular Design • Integrated Development Environment (IDE) • Database Management System (DBMS) • Design Review • Refinement	• [IEEE 1012] • [IEEE 1016] • [IEEE 12207] • [IEEE 42010]
Software Construction	• Detail Design Document • Pseudocode • Flow Chart • Program Code • Unit Test Plan • Integration Test Plan	• Detailed Design • Functional Programming • Object-Oriented Programming • IDE • DBMS • Black Box/White Box Testing • Basic Path Testing • Unit Testing • Code Review • Proof of Correctness • Software Reuse • Integration • Integration Testing	• [IEEE 1008] • [IEEE 1012] • [IEEE 1016] • [IEEE 12207]
Software Testing	• System Test Plan • Reliability Model • Software Maintenance Process	• Usability Testing • System Testing • Acceptance Testing • Regression Testing • Reliability Testing • Non Functional Software Testing	• [IEEE 829] • [IEEE 1012] • [IEEE 12207]
Software Maintenance	• Software Maintenance Process	• Automated Testing Tools • Maintenance Change • Impact Analysis • Inventory Analysis • Restructuring • Reverse Engineering • Re-engineering	• [IEEE 1219] • [IEEE 12207] • [IEEE 14764]

Software Metric

A software metric is a quantitative measure of the degree a software system, component, or process possesses a given attribute. Because of the abstract nature of software and special problems with software schedule, cost, and quality, data collection and the derived metrics are an essential part of software engineering. This is evidenced by the repeated reference to measurement and metrics in the SWEBOK. Table 2 describes software metrics that are collected and used in different areas of software development. As in Table 1 the list is not meant to be complete, but to illustrate the type and range of measures used in practice.

Table 2: Software Metrics * (SEBoK Original)

Category	Metrics
Management Metrics	• Size: Lines of Code (LOC*), Thousand Lines of Code (KLOC) • Size: Function points, Feature Points • Individual Effort: hours • Task Completion Time: hours, days, weeks • Project Effort: person-hours • Project Duration: months • Schedule: earned value • Risk Projection: risk description, risk likelihood, risk impact
Software Quality Metrics	• Defect Density - Defects/KLOC (e.g., for system test) • Defect Removal Rate – defect removed/hour (for review and test) • Test Coverage • Failure Rate
Software Requirements Metrics	• Change requests received, open, closed • Change request frequency • Effort required to implement a requirement change • Status of requirements traceability • User stories in the backlog
Software Design Metrics	• Cyclomatic Complexity • Weighted Methods per Class • Cohesion - Lack of Cohesion of Methods • Coupling - Coupling Between Object Classes • Inheritance - Depth of Inheritance Tree, Number of Children
Software Maintenance and Operation	• Mean Time Between Changes (MTBC) • Mean Time To Change (MTTC) • System Reliability • System Availability • Total Hours of Downtime

**Note:* Even though the LOC metric is widely used, using it comes with some problems and concerns: different languages, styles, and standards can lead to different LOC counts for the same functionality; there are a variety of ways to define and count LOC– source LOC, logical LOC, with or without comment lines, etc.; and automatic code generation has reduced the effort required to produce LOC.*

References

Works Cited

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.Swebok.org.

Dori, D. 2003. "Conceptual Modeling and System Architecting." *Communications of the ACM*, 46(10), pp. 62-65

[IEEE Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.Swebok.org.

[IEEE 828] IEEE Computer Society, IEEE Standard for *Computer Configuration Management in Systems and Software Engineering*, IEEE Std 828- 2012, 20012.

[IEEE 829] IEEE Computer Society, IEEE Standard for *Software and System Test Documentation*, IEEE Std 829-2008, 2008.

[IEEE 830] IEEE Computer Society, *Recommended Practice for Software Requirements Specifications*, IEEE Std 830-1998, 1998.

[IEEE 1008] IEEE Computer Society, IEEE Standard for *Software Unit* Testing, *IEEE Std 1008-1987, 1987.*

[IEEE 1012] — IEEE Computer Society, IEEE Standard for *System and Software Verification and Validation*. IEEE Std 1012-2002, 2012.

[IEEE 1016] — IEEE Computer Society, *Recommended Practice for Software Design Descriptions*. IEEE Std 1016-2002, 2002.

[IEEE 1058] IEEE Computer Society, IEEE Standard for *Software Project Plans*, IEEE Std 1058-1998, 1998.

[IEEE 1219] IEEE Computer Society, IEEE Standard for *Software Maintenance*, IEEE Std 1219-1998, 1998.

[IEEE 1540] IEEE Computer Society, IEEE Standard for *Risk Management*, IEEE Std 1540-2001, 2001.

[IEEE 12207] IEEE Computer Society, IEEE Standard for *Systems and Software Engineering —Software Life Cycle Processes*, IEEE Std 12207-2008. 2008.

[IEEE 14764] IEEE Computer Society, IEEE Standard for *Software Engineering - Software Life Cycle Processes - Maintenance*. IEEE Std 14764-2006. 2006

[IEEE 42010] IEEE Computer Society, IEEE Standard for *Systems and Software Engineering — Architecture Description*, IEEE Std 42010-2011, 2011.

Additional References

Chidamber, S.R., Kemerer, C.F. 1994. “A Metrics Suite for Object Oriented Design”, *IEEE Transactions on Software Engineering*. Vol. 20, No.6. June 1994.

Kan, Stephen H. 2003. *Metrics and Models in Software Quality Engineering*, 2nd edition. Addison-Wesley.

Li, M. and Smidts, C. 2003. “A Ranking of Software Engineering measures Based on Expert Opinion.” *IEEE Transactions on Software Engineering*. September 2003.

McConnell, Steve. 2009. *Code Complete*,. 2nf Ed. Microsoft Press.

Moore, James. 1997. *Software Engineering Standards: A User's Road Map*. Wiley-IEEE Computer Society Press.

Sommerville, I. 2010. *Software Engineering*. 9th Ed. Boston, MA, USA: Addison Wesley.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Management: Systems Engineering and Project Management

Systems Engineering and Project Management

The goal of project management is to plan and coordinate the work activities needed to deliver a satisfactory product, service, or enterprise endeavor within the constraints of schedule, budget, resources, infrastructure, and available staffing and technology. The purpose of this knowledge area (KA) is to acquaint systems engineers with the elements of project management and to explain the relationships between systems engineering (SE) and project management (PM).

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- The Nature of Project Management
- An Overview of the PMBOK® Guide
- Relationships between Systems Engineering and Project Management
- The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships
- Procurement and Acquisition

References

Works Cited

None.

Primary References

Fairley, R.E. 2009. *Managing and Leading Software Projects*. Hoboken, NJ, USA: John Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management*, 3rd ed. New York, NY, USA: John Wiley & Sons.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

The Nature of Project Management

While *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* provides an overview of project management for those seeking PMI certification, Fairley (2009) and Forsberg (2005) suggest another way to characterize the important aspects of project management:

- Planning and Estimating
- Measuring and Controlling
- Leading and Directing
- Managing Risk

Introduction

Project managers and systems engineers are both concerned with management issues such as planning, measuring and controlling, leading, directing, and managing risk. In the case of project managers, the project attributes to be managed include project plans; estimates; schedule; budget; project structure; staffing; resources; infrastructure; and risk factors. Product attributes managed by systems engineers include items such as requirements allocation and flow-down; system architecture; structure of and interactions among technical teams; specialty engineering; integration; verification; and validation.

The exact allocation of the SE and PM duties depend on many factors, such as customer and stakeholder interactions, organizational structure of the parent organization, and relationships with affiliate contractors and subcontractors. (See the article on The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships in this KA.)

Planning and Estimating

Planning

Planning a project involves providing answers to the who, what, where, when, and why of every project:

- **Who:** Addresses staffing issues (competencies, numbers of staff, communication and coordination)
- **What:** Addresses the scope of activities
- **Where:** Addresses issues of locale (local, geographically distributed)
- **When:** Addresses scheduling issues
- **Why:** Addresses rationale for conducting a project

Guidance for developing project plans can be found in INCOSE (2012), NASA (2007), and ISO/IEC/IEEE Standard 16326:2009. It is often observed that communication and coordination among stakeholders during project planning are equally as important as (and sometimes more important than) the documented plan that is produced.

In defense work, event-driven integrated master plans and time-driven integrated master schedules are planning products. Chapter 11 of the Defense Acquisition Guidebook provides details (DAU 2010).

Estimating

Estimation is an important element of planning. An estimate is a projection from past to future, adjusted to account for differences between past and future. Estimation techniques include analogy, rule of thumb, expert judgment, and use of parametric models such as the PRICE model for hardware, COCOMO for software projects and COSYSMO for systems projects (Stewart 1990; Boehm et al. 2000; Valerdi 2008).

Entities estimated include (but are not limited to) schedule; cost; performance; and risk.

Systems engineering contributes to project estimation efforts by ensuring that

- the overall system life cycle is understood;
- dependencies on other systems and organizations are identified;
- the logical dependencies during development are identified; and
- resources and key skills are identified and planned.

Additionally, high-level system architecture and risk assessment provide the basis for both the work breakdown structure and the organizational breakdown structure.

Measuring and Controlling

Measuring and controlling are the key elements of executing a project. Measurement includes collecting measures for work products and work processes. For example, determining the level of coverage of requirements in a design specification can be assessed through review, analysis, prototyping, and traceability. Effort and schedule expended on the work processes can be measured and compared to estimates; earned value tracking can be used for this purpose. Controlling is concerned with analyzing measurement data and implementing corrective actions when actual status does not align with planned status.

Systems engineers may be responsible for managing all technical aspects of project execution, or they may serve as staff support for the project manager or project management office. Organizational relationships between systems engineers and project managers are presented in Team Capability. Other organizational considerations for the relationships between systems engineering and project management are covered in the Enabling Systems Engineering knowledge area.

Additional information on measurement and control of technical factors can be found in the Measurement and Assessment and Control articles in Part 3: Systems Engineering and Management.

Leading and Directing

Leading and directing requires communication and coordination among all project stakeholders, both internal and external. Systems engineers may be responsible for managing all technical aspects of project execution, or they may serve as staff support for the project manager or project management office. Organizational relationships between systems engineers and project managers are presented in the article Team Capability in Part 5. Other organizational considerations for the relationships between systems engineering and project management are discussed in Part 5: Enabling Systems Engineering.

Managing Risk

Risk management is concerned with identifying and mitigating potential problems before they become real problems. Systems engineering projects are, by nature, high-risk endeavors because of the many unknowns and uncertainties that are inherent in projects. Because new risk factors typically emerge during a project, ongoing continuous risk management is an important activity for both systems engineers and project managers.

Potential and actual problems may exist within every aspect of a project. Systems engineers are typically concerned with technical risk and project managers with programmatic risk. Sometimes, technical risk factors are identified and confronted by systems engineers and programmatic risk factors are identified and confronted by project managers without adequate communication between them. In these cases, appropriate tradeoffs among requirements, schedule, budget, infrastructure, and technology may not be made, which creates additional risk for the successful outcome of a project.

In the last ten years, there has been an increasing interest in opportunity management as the converse of risk management. Hillson(2003), Olsson (2007), and Chapman and Ward (2003) provide highly cited introductions.

Additional information on risk management for systems engineering projects can be found in the Risk Management article in Part 3: Systems Engineering and Management.

References

Works Cited

Boehm, B., C. Abts., A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. 2000. *Software Cost Estimation with COCOMO II.* Upper Saddle River, NJ, USA: Prentice Hall.

Chapman, C., and S. Ward. 2003. *Project Risk Management: Processes, Techniques and Insights.* Chichester, West Sussex, England, UK: John Wiley & Sons.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken NJ, USA: John Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* Hoboken, NJ, USA: John Wiley & Sons.

Hillson, David. 2003. *Effective Opportunity Management for Projects: Exploiting Positive Risk.* Boca Raton, FL, USA: CRC Press.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2009. ISO/IEC/IEEE 16326:2009(E). *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE).

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration.

Olsson, Rolf. 2007. "In search of opportunity management: Is the risk management process enough?" *International Journal of Project Management*, 25 (8), 745–752, 2011.

Stewart, Rodney. 1990. *Cost Estimating.* New York, NY, USA: Wiley.

Valerdi, R. *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort.* Saarbrucken, Germany: VDM Verlag.

Primary References

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, NJ, USA: John Wiley & Sons.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide).* 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

Blanchard, B. 2008. *System Engineering Management.* Hoboken, NJ, USA: John Wiley & Sons.

Kerzner, Harold. 2003. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 8th ed. Hoboken, NJ, USA: John Wiley & Sons.

Martin, J. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products.* London, UK: Taylor and Francis Group CRC-Press, LLC.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

An Overview of the PMBOK® Guide

The *Guide to the Project Management Book of Knowledge (PMBOK® Guide)* is published and maintained by the Project Management Institute (PMI). It is acknowledged as the authoritative documentation of good practices in project management. It is also the basis for certification exams to qualify Project Management Professionals (PMPs). Many organizations require PMP certification as a basic qualification for the role of project manager.

Overview

According to Section 1.3 of the *PMBOK® Guide*, project management is *accomplished through the appropriate application and integration of the 47 logically grouped project management processes, which are categorized into five Process Groups* (PMI 2013). The five Process Groups are

1. Initiating Process Group
2. Planning Process Group
3. Executing Process Group
4. Monitoring and Controlling Process Group
5. Closing Process Group

Each of the 47 processes is specified by Inputs, Tools & Techniques, and Outputs. Data flow diagrams are used in the PMBOK to illustrate the relationships between each process and the other processes in which each process interacts. The processes are also grouped into ten Knowledge Areas. These Knowledge Areas are

1. Project Integration Management
2. Project Scope Management
3. Project Time Management
4. Project Cost Management
5. Project Quality Management
6. Project Human Resources Management
7. Project Communications Management
8. Project Risk Management
9. Project Procurement Management
10. Project Stakeholder Management

The five process groups are discussed in more detail next.

Initiating Process Group

Activities performed in the **Initiating** process group include obtaining authorization to start a project; defining the high-level scope of the project; developing and obtaining approval for the project charter; performing key stakeholder analysis; and identifying and documenting high-level risks, assumptions, and constraints. The **Initiating** process group contains two processes: develop the project charter and identify stakeholders.

Planning Process Group

The **Planning** process group consists of 24 processes, including assessing detailed project requirements, constraints, and assumptions with stakeholders; developing the project management plan; creating the work breakdown structure; developing a project schedule; determining a project budget; and planning for quality management, human resource management, communication management, change and risk management, procurement management, and stakeholder management. The integrated project management plan is presented to key stakeholders.

Executing Process Group

The **Executing** process group includes eight processes that involve performing the work necessary to achieve the stated objectives of the project. Activities include obtaining and managing project resources; executing the tasks defined in the project plan; implementing approved changes according to the change management plan; performing quality assurance; acquiring, developing, and managing the project team; managing communications; conducting procurements; and managing stakeholder engagement.

Monitoring and Controlling Process Group

The **Monitoring and Controlling** process group is comprised of 11 processes that include validate and control scope; control schedule; control cost; control quality; control communications, control risks; control procurements; and control stakeholder engagement. Activities include measuring project performance and using appropriate tools and techniques; managing changes to the project scope, schedule, and costs; ensuring that project deliverables conform to quality standards; updating the risk register and risk response plan; assessing corrective actions on the issues register; and communicating project status to stakeholders.

Closing Process Group

The **Closing** process group involves two processes: closing project or phase and closing procurements. Closing the project or phase involves finalizing all project activities, archiving documents, obtaining acceptance for deliverables, and communicating project closure. Other activities include transferring ownership of deliverables; obtaining financial, legal, and administrative closure; distributing the final project report; collating lessons learned; archiving project documents and materials; and measuring customer satisfaction.

The scope of project management, as specified in the PMBOK Guide, encompasses the total set of management concerns that contribute to successful project outcomes.

References

Works Cited

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

Blanchard, B. 2008. *System Engineering Management*. Hoboken, NJ, USA: John Wiley & Sons.

Fairley, R.E. 2009. *Managing and Leading Software Projects*. Hoboken, NJ, USA: John Wiley & Sons.

Martin, J. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products*. London, UK: Taylor and Francis Group CRC-Press, LLC.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Relationships between Systems Engineering and Project Management

This topic discusses the relationship between systems engineering (SE) and project management (PM). As with software engineering, there is a great deal of overlap. Depending on the environment and organization, the two disciplines can be disjoint, partially intersecting, or one can be seen as a subset of the other. While there is no standard relationship, the project manager and the systems engineer encompass the technical and managerial leadership of a project between them, which requires the enterprise of each project manager and system engineer to work out the particular details for their own context.

Overlap

There is a great deal of significant overlap between the scope of systems engineering, as described here (in the SEBoK), CMMI (2011), and other resources and the scope of project management, as described in the *PMBOK® Guide* (PMI 2013), CMMI (2011), and other resources as illustrated in Figure 1.

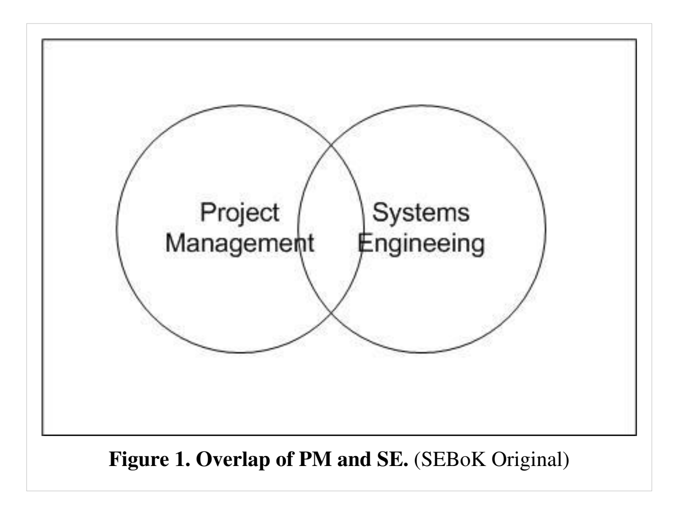


Figure 1. Overlap of PM and SE. (SEBoK Original)

These sources describe the importance of understanding the scope of the work at hand, how to plan for critical activities, how to manage efforts while reducing risk, and how to successfully deliver value to a customer. The systems engineer working on a project will plan, monitor, confront risk, and deliver the technical aspects of the project, while the project manager is concerned with the same kinds of activities for the overall project. Because of these shared concerns, at times there may be confusion and tension between the roles of the project manager and the systems engineer on a given project. As shown in Figure 2, on some projects, there is no overlap in responsibility. On other projects, there may be shared responsibilities for planning and managing activities. In some cases, particularly for smaller projects, the project manager may also be the lead technical member of the team performing both roles of project manager and systems engineer.

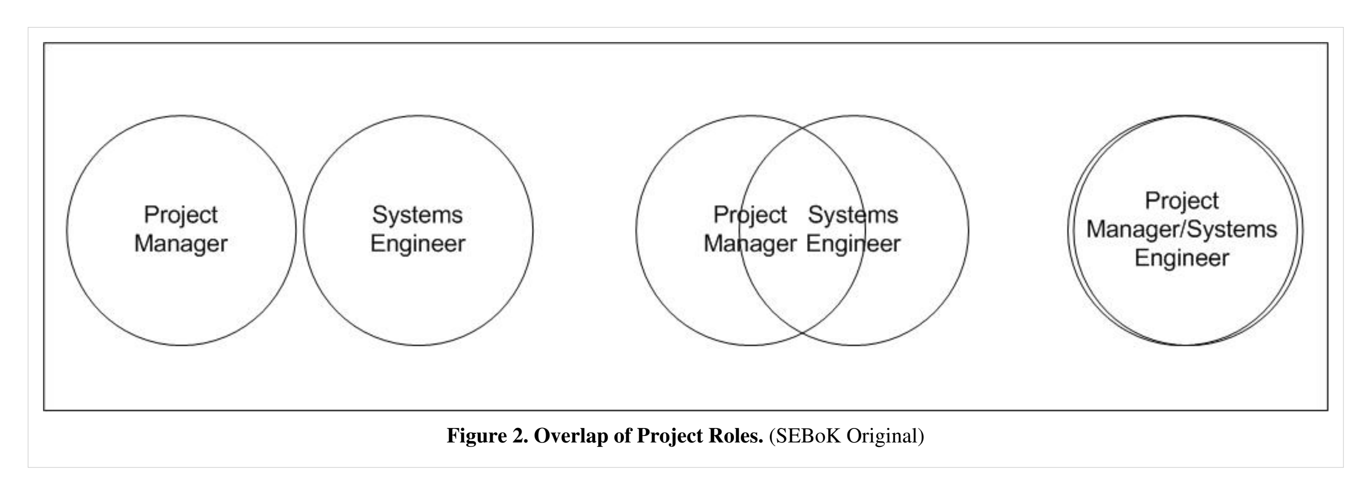


Figure 2. Overlap of Project Roles. (SEBoK Original)

Defining Roles and Responsibilities

Regardless of how the roles are divided up on a given project, the best way to reduce confusion is to explicitly describe the roles and responsibilities of the project manager and the systems engineer, as well as other key team members. The Project Management Plan (PMP) and the Systems Engineering Management Plan (SEMP) are key documents used to define the processes and methodologies the project will employ to build and deliver a product or service.

The PMP is the master planning document for the project. It describes all activities, including technical activities, to be integrated and controlled during the life of the program. The SEMP is the master planning document for the systems engineering technical elements. It defines SE processes and methodologies used on the project and the relationship of SE activities to other project activities. The SEMP must be consistent with, and evolve in concert, with the PMP. In addition, some customers have technical management plans and expectations that the project's SEMP integrate with customer plans and activities. In the U.S. Department of Defense, most government project teams have a systems engineering plan (SEP) with an expectation that the contractor's SEMP will integrate and remain consistent with customer technical activities. In cases where the project is developing a component of a larger system, the component project's SEMP will need to integrate with the overall project's SEMP.

Given the importance of planning and managing the technical aspects of the project, an effective systems engineer will need to have a strong foundation in management skills and prior experience, as well as possess strong technical depth. From developing and defending basis of estimates, planning and monitoring technical activities, identifying and mitigating technical risk, and identifying and including relevant stakeholders during the life of the project, the systems engineer becomes a key member of the project's management and leadership team. Additional information on Systems Engineering Management and Stakeholder Needs and Requirements can be found in Part 3: Systems Engineering and Management.

Practical Considerations

Effective communication between the project manager and the system engineer is essential for mission accomplishment. This communication needs to be established early, and occur frequently.

Resource reallocation, schedule changes, product/system changes and impacts, risk changes: all these and more need to be quickly and clearly discussed between the PM and SE.

References

Works Cited

CMMI. 2011. *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Old Tappan, NJ, USA: Pearson Education.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

Chrissis, M.B, M. Konrad, S. Shrum. 2011. *CMMI for Development: Guidelines for Process Integration and Product Improvement*, 3rd ed. Boston, MA, USA: Addison-Wesley Professional.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

NASA. 2007. *Systems Engineering Handbook*, Revision 1. Washington, DC, USA: National Aeronautics and Space Administration (NASA). NASA/SP-2007-6105.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships

This article reviews various project structures that impact or provide governance to the project and that require key involvement from the program manager and the systems engineer. These structures include: the structure of the organization itself (functional, project, matrix, and specialized teams, such as Integrated Product Teams (IPTs), Change Control Boards (CCBs), and Engineering Review Boards (ERBs). This article also addresses the influence of schedule-driven versus requirements-driven projects on these structures.

The Relationships between Systems Engineering and Project Management is covered in a related article.

An Overview of Project Structures

Project management and systems engineering governance are dependent on the organization's structure. For some projects, systems engineering is subordinated to project management and in other cases, project management provides support to systems engineering. These alternatives are illustrated in Figures 1 and 2 of the Organizing the Team section in Team Capability.

A project exists within the structural model of an organization. Projects are one-time, transient events that are initiated to accomplish a specific purpose and are terminated when the project objectives are achieved. Sometimes, on small projects, the same person accomplishes the work activities of both project management and systems engineering. Because the nature of the work activities are significantly different, it is sometimes more effective to have two persons performing project management and systems engineering, each on a part-time basis. On larger projects there are typically too many tasks to be accomplished for one person to accomplish all of the necessary work. Very large projects may have project management and systems engineering offices with a designated project manager and a designated lead systems engineer.

Projects are typically organized in one of three ways: (1) by functional structure, (2) by project structure, and (3) by a matrix structure (see Systems Engineering Organizational Strategy for a fourth structure and related discussion). In a function-structured organization, workers are grouped by the functions they perform. The systems engineering functions can be: (1) distributed among some of the functional organizations, (2) centralized within one organization or (3) a hybrid, with some of the functions being distributed to the projects, others centralized and others are distributed to functional organization. The following figure provides an organizational structure continuum and

illustrates levels of governance among the functional organizations and the project.

- In a functional-structured organization, the project manager is a coordinator and typically has only limited control over the systems engineering functions. In this type of organization, the functional manager typically controls the project budget and has authority over the project resources. However, the organization may or may not have a functional unit for systems engineering. In the case where there is a functional unit for systems engineering, systems engineers are assigned across existing projects. Trades can be made among their projects to move the priority of a specific systems engineering project ahead of other projects; thus, reducing the nominal schedule for that selected project. However, in the case where there is not a functional unit for systems engineering, the project manager may have to find alternate sources of staffing for systems engineering – for example, hiring systems engineering talent or consultants, or may consider promoting or expanding the responsibilities of a current team member, etc.
- In a project-structured organization, the project manager has full authority and responsibility for managing the budget and resources to meet the schedule requirements. The systems engineer is subject to the direction of the project manager. The project manager may work with human resources or a personnel manager or may go outside the organization to staff the project.
- Matrix-structured organization can have the advantages of both the functional and project structures. For a schedule driven project, function specialists are assigned to projects as needed to work for the project manager to apply their expertise on the project. Once they are no longer needed, they are returned to their functional groups (e.g. home office). In a weak matrix, the functional managers have authority to assign workers to projects and project managers must accept the workers assigned to them. In a strong matrix, the project manager controls the project budget and can reject workers from functional groups and hire outside workers if functional groups do not have sufficient available and trained workers.

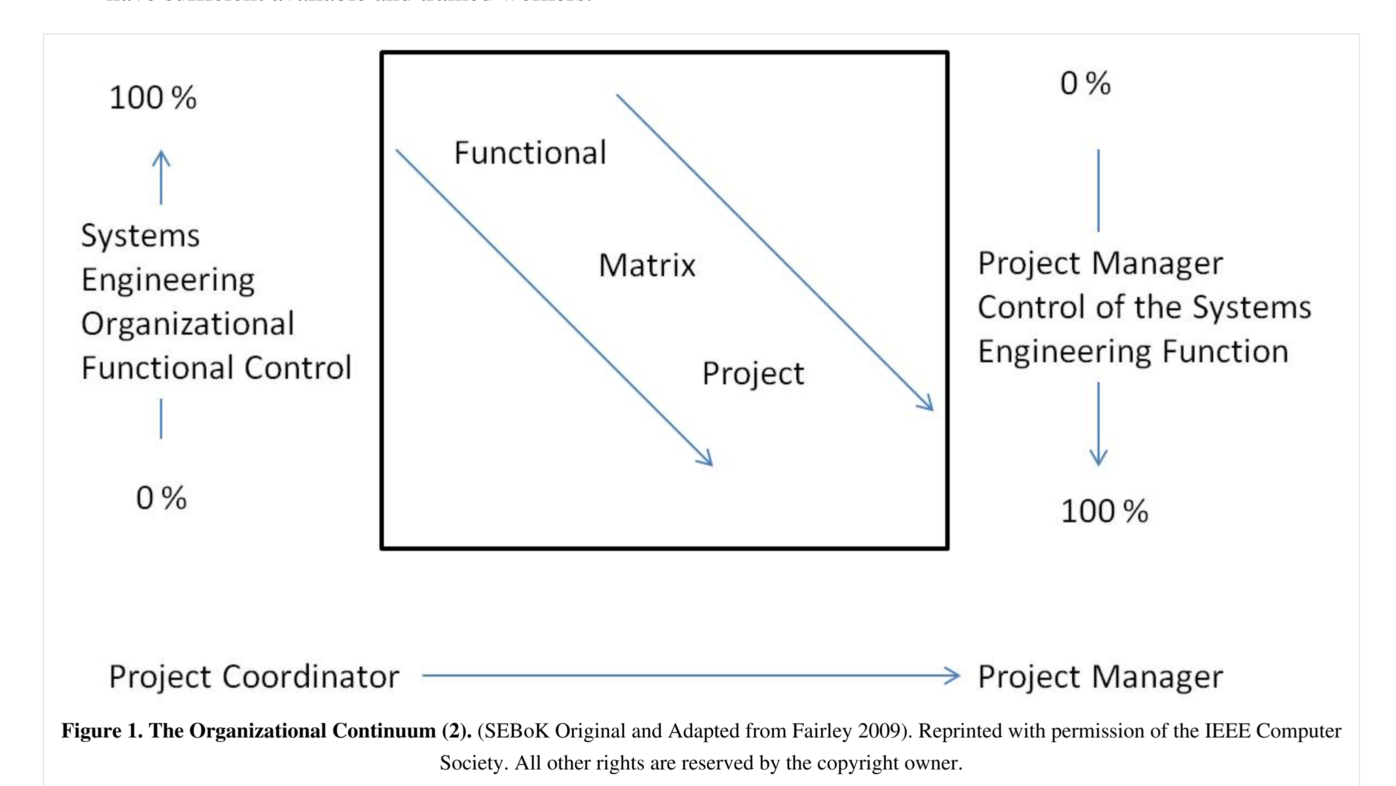


Figure 1. The Organizational Continuum (2). (SEBoK Original and Adapted from Fairley 2009). Reprinted with permission of the IEEE Computer Society. All other rights are reserved by the copyright owner.

In all cases, it is essential that the organizational and governance relationships be clarified and communicated to all project stakeholders and that the project manager and systems engineer work together in a collegial manner.

The Project Management Office (PMO) provides centralized control for a set of projects. The PMO is focused on meeting the business objectives leveraging a set of projects, while the project managers are focused on meeting the objectives of those projects that fall under their purview. PMOs typically manage shared resources and coordinate

communication across the projects, provide oversight and manage interdependencies, and drive project-related policies, standards, and processes. The PMO may also provide training and monitor compliance (PMI 2013).

Schedule-Driven versus Requirements-Driven Influences on Structure and Governance

This article addresses the influences on governance relationships between the project manager and the systems engineer. One factor that establishes this relationship is whether a project is schedule-driven or requirements-driven.

In general, a project manager is responsible for delivering an acceptable product/service on the specified delivery date and within the constraints of the specified schedule, budget, resources, and technology.

The systems engineer is responsible for collecting and defining the operational requirements, specifying the systems requirements, developing the system design, coordinating component development teams, integrating the system components as they become available, verifying that the system to be delivered is correct, complete and consistent to its technical specification, and validating the operation of the system in its intended environment.

From a governance perspective, the project manager is often thought of as being a movie producer who is responsible for balancing the schedule, budget, and resource constraints to meet customer satisfaction. The systems engineer is responsible for product content; ergo, the systems engineer is analogous to a movie director.

Organizational structures, discussed previously, provide the project manager and systems engineer with different levels of governance authority. In addition, schedule and requirements constraints can influence governance relationships. A schedule-driven project is one for which meeting the project schedule is more important than satisfying all of the project requirements; in these cases lower priority requirements may not be implemented in order to meet the schedule.

Classic examples of these types of projects are:

- a project that has an external customer with a contractual delivery date and an escalating late delivery penalty, and
- a project for which delivery of the system must meet a major milestone (e.g. a project for an announced product release of a cell phone that is driven by market considerations).

For schedule-driven projects, the project manager is responsible for planning and coordinating the work activities and resources for the project so that the team can accomplish the work in a coordinated manner to meet the schedule. The systems engineer works with the project manager to determine the technical approach that will meet the schedule. An Integrated Master Schedule (IMS) is often used to coordinate the project.

A requirements-driven project is one for which satisfaction of the requirements is more important than the schedule constraint. Classic examples of these types of projects are:

1. exploratory development of a new system that is needed to mitigate a potential threat (e.g. military research project) and
2. projects that must conform to government regulations in order for the delivered system to be safely operated (e.g., aviation and medical device regulations).

An Integrated Master Plan is often used to coordinate event-driven projects.

To satisfy the product requirements, the systems engineer is responsible for the making technical decisions and making the appropriate technical trades. When the trade space includes cost, schedule, or resources, the systems engineer interacts with the project manager who is responsible for providing the resources and facilities needed to implement a system that satisfies the technical requirements.

Schedule-driven projects are more likely to have a management structure in which the project manager plays the central role, as depicted in Figure 1 of the Organizing the Team section in Team Capability. Requirement-driven projects are more likely to have a management structure in which the systems engineer plays the central role, as depicted in Figure 2 of the Organizing the Team section in Team Capability.

Along with the Project Management Plan and the Systems Engineering Management Plan, IMP/IMS are critical to this process.

Related Structures

Integrated Product Teams (IPTs), Change Control Boards (CCBs), and Engineering Review Boards (ERBs) are primary examples of project structures that play a significant role in project governance and require coordination between the project manager, systems engineer and other members of the team.

Integrated Product Team

The Integrated Product Team (IPT) ensures open communication flow between the government and industry representatives as well as between the various product groups (see Good Practices in Planning). There is typically a top level IPT, sometimes referred to as the Systems Engineering and Integration Team (SEIT) (see Systems Engineering Organizational Strategy), that oversees the lower level IPTs. The SEIT can be led by either the project manager for a specific project or by the systems engineering functional manager or functional lead across many projects. Each IPT consists of representatives from the appropriate management and technical teams that need to collaborate on systems engineering, project management, and other activities to create a high quality product. These representatives meet regularly to ensure that the technical requirements are understood and properly implemented in the design. Also see Team Capability.

Change Control Board

An effective systems engineering approach includes a disciplined process for change control as part of the larger goal of configuration management. The primary objective of configuration management is to track changes to project artifacts that include software, hardware, plans, requirements, designs, tests, and documentation. Alternatively, a Change Control Board (CCB) with representatives from appropriate areas of the project is set up to effectively analyze, control and manage changes being proposed to the project. The CCB typically receives an Engineering Change Proposal (ECP) from design/development, production, or operations/support and initially reviews the change for feasibility. The ECP may also be an output of the Engineering Review Board (ERB) (see next section). If determined feasible, the CCB ensures there is an acceptable change implementation plan and proper modification and installation procedures to support production and operations.

There may be multiple CCBs in a large project. CCBs may be comprised of members from both the customer and the supplier. As with the IPTs, there can be multiple levels of CCB starting with a top level CCB with CCBs also existing at the subsystem levels. A technical lead typically chairs the CCB; however, the board includes representation from project management since the CCB decisions will have an impact on schedule, budget, and resources.

See Figure 2 under Configuration Management for a flow of the change control process adapted from Blanchard and Fabrycky (2011). See also Capability Updates, Upgrades, and Modernization, and topics included under Enabling Teams. See also the UK West Coast Route Modernisation Project which provides an example where change control was an important success factor.

Engineering Review Board

Another example of a board that requires collaboration between technical and management is the Engineering Review Board (ERB). Examples of ERBs include the Management Safety Review Board (MSRB) (see Safety Engineering. Responsibilities of the ERB may include technical impact analysis of pending change requests (like the CCB), adjudication of results of engineering trade studies, and review of changes to the project baseline. In some cases the ERB may be the management review board and the CCB may be the technical review board. Alternatively, in a requirement driven organization the ERB may have more influence while in a schedule driven organization the

CCB may have more impact.

References

Works Cited

Blanchard, B.S., and W. J. Fabrycky. 2011. *Systems Engineering and Analysis.* 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* IEEE Computer Society, John Wiley & Sons, Inc. Publication. ISBN: 978-0-470-29455-0.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* 3rd ed. New York, NY, USA: John Wiley & Sons.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Procurement and Acquisition

Procurement is the act of buying goods and services. Acquisition covers the conceptualization, initiation, design, development, testing, contracting, production, deployment, logistics support, modification, and disposal of weapons and other systems, as well as supplies or services (including construction) to satisfy organizational needs intended for use in, or in support of, defined missions (DAU 2010; DoD 2001).

Acquisition covers a much broader range of topics than procurement. Acquisition spans the whole life cycle of acquired systems. The procurement of appropriate systems engineering (SE) acquisition activities and levels of SE support is critical for an organization to meet the challenge of developing and maintaining complex systems.

The *Guide for Integrating Systems Engineering into DoD Acquisition Contracts* addresses how systems engineering activities are integrated into the various elements of acquisition and procurement (DoD 2006a).

Acquisition Process Model

Multiple acquisition process models exist. An acquisition process for major systems in industry and defense is shown in Figure 1. The process of acquisition is defined by a series of phases during which technology is defined and matured into viable concepts. These concepts are subsequently developed and readied for production, after which the systems produced are supported in the field.

Acquisition planning is the process of identifying and describing needs, capabilities, and requirements, as well as determining the best method for meeting those requirements (e.g., program acquisition strategy). This process includes procurement; thus, procurement is directly linked to the acquisition process model. The process model present in Figure 1 allows a given acquisition to enter the process at any of the development phases.

For example, a system using unproven technology would enter at the beginning stages of the process and would proceed through a lengthy period of technology maturation. On the other hand, a system based on mature and proven technologies might enter directly into engineering development or sometimes even production.

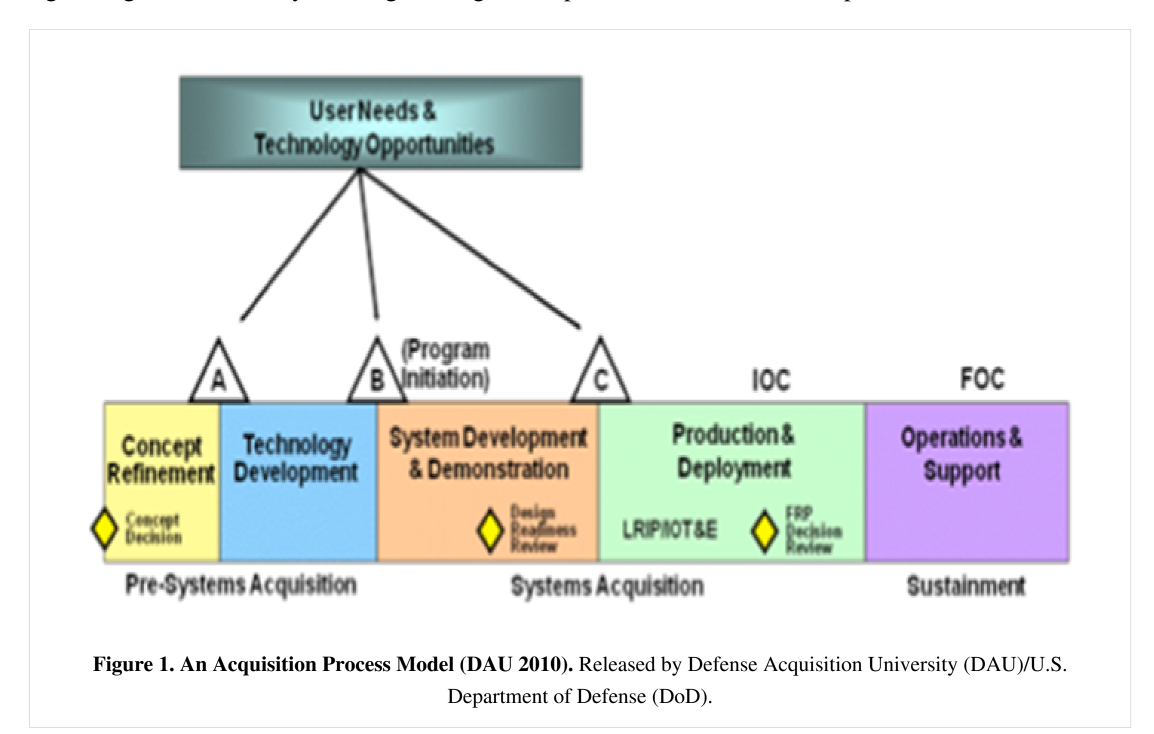


Figure 1. An Acquisition Process Model (DAU 2010). Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Systems Engineering Role in the Acquisition Process

The procurement of complex systems usually requires a close relationship between the offeror and supplier SE teams due to the breadth and depth of SE activities. SE is an overarching process that the program team applies in order to transition from a stated capability need to an affordable, operationally effective, and suitable system.

SE is important to every phase of the acquisition process. SE encompasses the application of SE processes across the acquisition life cycle and is intended to be an integrating mechanism for balanced solutions addressing capability needs, design considerations, and constraints. It is also intended to address limitations imposed by technology, budget, and schedule.

SE is an interdisciplinary approach; that is, it is a structured, disciplined, and documented technical effort to simultaneously design and develop system products and processes to satisfy the needs of the customer. Regardless of the scope and type of program, or at what point it enters the program acquisition life cycle, the technical approach to the program needs to be integrated with the acquisition strategy to obtain the best program solution.

Acquisition and procurement in the commercial sector have many characteristics in common with their counterparts in the realm of government contracting, although the processes in the commercial world are usually accomplished with fewer rigors than occur between government and contractor interactions. Offshore outsourcing is commonly practiced in the commercial software arena with the goal of reducing the cost of labor. Commercial organizations sometimes subcontract with other commercial organizations to provide missing expertise and to balance the ebb and flow of staffing needs.

In some cases, relations between the contracting organization and the subcontractor are strained because of the contracting organization's desire to protect its intellectual property and development practices from potential exposure to the subcontractor. Commercial organizations often have lists of approved vendors that are used to expedite the procurement of needed equipment, products, and services. In these situations, commercial organizations have processes to evaluate and approve vendors in ways that are analogous to the qualification of government contractors. Many commercial organizations apply SE principles and procedures even though they may not identify

the personnel and job functions as “systems engineers” or “systems engineering.”

Importance of the Acquisition Strategy in the Procurement Process

The acquisition strategy is usually developed during the front end of the acquisition life cycle. (For an example of this, see the Technology Development Phase in Figure 1.) The acquisition strategy provides the integrated strategy for all aspects of the acquisition program throughout the program life cycle.

In essence, the acquisition strategy is a high-level business and technical management approach designed to achieve program objectives within specified resource constraints. It acts as the framework for planning, organizing, staffing, controlling, and leading a program, as well as for establishing the appropriate contract mechanisms. It provides a master schedule for research, development, testing, production, fielding, and other SE related activities essential for program success, as well as for formulating functional strategies and plans.

The offeror’s program team, including systems engineering, is responsible for developing and documenting the acquisition strategy, which conveys the program objectives, direction, and means of control based on the integration of strategic, technical, and resource concerns. A primary goal of the acquisition strategy is the development of a plan that will minimize the time and cost of satisfying an identified, validated need while remaining consistent with common sense and sound business practices. While the contract officer (CO) is responsible for all contracting aspects, including determining which type of contract is most appropriate, and following the requirements of existing regulations, directives, instructions, and policy memos of an organization, the program manager (PM) works with the CO to develop the best contract/procurement strategy and contract types.

Relating Acquisition to Request for Proposal and Technical Attributes

There are several formats for requesting proposals from offerors for building complex systems. Figure 2 relates acquisition program elements to a representative request for proposal (RFP) topical outline and key program technical attributes that have been used by the Department of Defense. In general, programs have a better chance of success when both the offeror and supplier understand the technical nature of the program and the need for the associated SE activities.

The offeror and supplier need to clearly communicate the technical aspect of the program throughout the procurement process. The offeror’s RFP and the associated supplier proposal represent one of the formal communications paths. A partial list of key program technical attributes is presented in Figure 2.

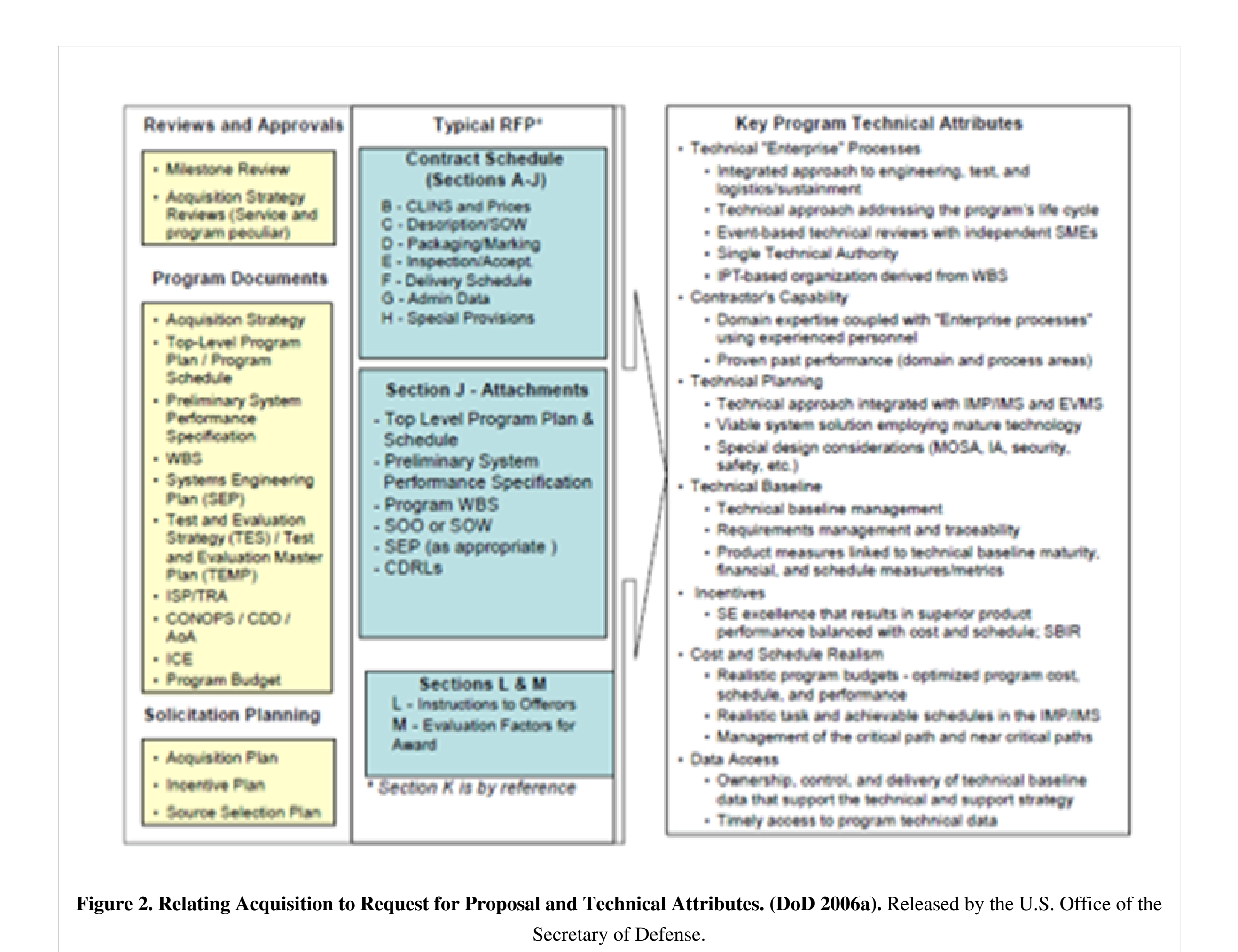


Figure 2. Relating Acquisition to Request for Proposal and Technical Attributes. (DoD 2006a). Released by the U.S. Office of the Secretary of Defense.

Contract-Related Activities and the Offeror's Systems Engineering and Project Management Roles

A clear understanding of the technical requirements is enhanced via the development of a Systems Engineering Plan (SEP). The SEP documents the system engineering strategy for a project or program and acts as the blueprint for the conduct, management, and control of the technical aspects of the acquisition program (DoD 2011). The SEP documents the SE structure and addresses government and contractor boundaries. It also summarizes the program's selected acquisition strategy. It identifies and links to program risks. It also describes how the contractor's, and sometimes the subcontractor's and suppliers', technical efforts are to be managed.

Once the technical requirements are understood, a contract may be developed and followed by the solicitation of suppliers. The offeror's PM, chief or lead systems engineer, and CO must work together to translate the program's acquisition strategy and associated technical approach (usually defined in a SEP) into a cohesive, executable contract(s).

Table 1 shows some key contracting-related tasks with indicators of the roles of the PM and LSE.

Table 1. Offeror's Systems Engineering and Program Management Roles (DoD 2006). Released by the U.S. Office of the Secretary of Defense.

Typical Contract-Related Activities	System Engineer and Project Manager Roles
1. Identify overall procurement requirements and associated budget. Describe the offer's needs and any constraints on the procurement.	Lead system engineer (LSE) provides program technical requirements. PM provides any programmatic related requirements.
2. Identify technical actions required to successfully complete technical and procurement milestones. The program's SEP is the key source for capturing this technical planning.	LSE defines the technical strategy/approach and required technical efforts. This should be consistent with the program's Acquisition Strategy.
3. Document market research results and identify potential industry sources.	PM and LSE identify programmatic and technical information needed and assist in evaluating the results.
4. Prepare a Purchase Request, including product descriptions; priorities, allocations and allotments; architecture; government-furnished property or equipment (or Government-Off-The-Shelf (GOTS); government-furnished information; information assurance and security considerations; and required delivery schedules.	PM and LSE ensure the specific programmatic and technical needs are defined clearly (e.g., commercial-off-the-shelf (COTS) products).
5. Identify acquisition streamlining approach and requirements, budgeting and funding, management information requirements, environmental considerations, offeror's expected skill sets, and milestones. These should be addressed in the Acquisition Strategy.	The procurement team work together, but the CO has the prime responsibility. The PM is the owner of the program Acquisition Strategy. The LSE develops and reviews (and the PM approves) the technical strategy.
6. Plan the requirements for the contract Statement of Objectives (SOO) / Statement of Work (SOW) / specification, project technical reviews, acceptance requirements, and schedule.	LSE is responsible for the development of the technical aspects of the SOO/SOW.
7. Plan and conduct Industry Days as appropriate.	PM and LSE supports the CO in planning the meeting agenda to ensure technical needs are discussed.
8. Establish contract cost, schedule, and performance reporting requirements. Determine an incentive strategy and appropriate mechanism (e.g., Award Fee Plan and criteria).	LSE provides technical resource estimates. LSE supports development of the Work Breakdown Structure (WBS) based on preliminary system specifications, determines event-driven criteria for key technical reviews, and determines what technical artifacts are baselined. The PM and LSE advise the CO in developing the metrics/criteria for an incentive mechanism.
9. Identify data requirements.	LSE identifies all technical Contractor Data Requirements List (CDRL) and technical performance expectations.
10. Establish warranty requirements, if applicable.	LSE works with the CO to determine cost-effective warranty requirements.
11. Prepare a Source Selection Plan (SSP) and RFP (for competitive contracts).	PM and LSE provide input to the SSP per the SOO/SOW.
12. Conduct source selection and award the contract to the successful offeror.	PM and LSE participate on evaluation teams.

Offeror and Supplier Interactions

There should be an environment of open communication prior to the formal source selection process. This ensures that the supplier understands the offeror's requirements and that the offeror understands the supplier's capabilities and limitations, as well as enhancing the supplier's involvement in the development of a program acquisition strategy. During the pre-solicitation phase, the offeror develops the solicitation and may ask suppliers to provide important insights into the technical challenges, program technical approach, and key business motivations.

For example, potential bidders could be asked for their assessment of a proposed system's performance based on the maturity level of new and existing technologies.

Contracts and Subcontracts

Typical types of contracts include the following:

- **Fixed Price**: In a fixed price contract the offeror proposes a single price for all products and services to implement the project. This single price is sometimes referred to as low bid or lump sum. A fixed price contract transfers the project risks to the supplier. When there is a cost overrun, the supplier absorbs it. If the supplier performs better than planned, their profit is higher. Since all risks are absorbed by the supplier, a fixed price bid may be higher to reflect this.
- **Cost-reimbursement** [Cost plus]: In a cost-reimbursement contract the offeror provides a fixed fee, but also reimburses the contractor for labor, material, overhead, and administration costs. Cost-reimbursement type contracts are used when there is a high level of project risk and uncertainty. With this type of contract the risks reside primarily with the offeror. The supplier gets reimbursed for all of its costs. Additional costs that arise due to changes or rework are covered by the offeror. This type of contract is often recommended for the system definition of hardware and software development when there is a risk of stakeholder changes to the system.
- **Subcontracts**: A subcontractor performs work for another company as part of a larger project. A subcontractor is hired by a general contractor (also known as a prime or main contractor) to perform a specific set of tasks as part of the overall project. The incentive to hire subcontractors is either to reduce costs or to mitigate project risks. The systems engineering team is involved in establishing the technical contract requirements, technical selection criteria, acceptance requirements, and the technical monitoring and control processes.
- **Outsource contracts**: Outsourced contracts are used to obtain goods or services by contracting with an outside supplier. Outsourcing usually involves contracting a business function, such as software design and code development, to an external provider.
- **Exclusively Commercial Off-the-Shelf (COTS)**: Exclusively COTS contracts are completely satisfied with commercial solutions that require no modification for use. COTS solutions are used in the environment without modifying the COTS system. They are integrated into an existing user's platform or integrated into an existing operational environment. The systems engineering team is involved in establishing the technical contract requirements, technical acceptance, and technical selection criteria.
- **Integrated COTS**: Integrated COTS contracts use commercially available products and integrate them into existing user platforms or operational environments. In some cases, integrated COTS solutions modify the system's solution. The cost of integrating the commercial COTS product into the operational environment can exceed the cost of the COTS product itself. As a result, the systems engineering team is usually involved in establishing the technical outsourcing contract requirements, technical selection criteria, technical monitoring and control processes, and technical acceptance and integration processes.
- **COTS Modification**: COTS modification requires the most time and cost because of the additional work needed to modify the COTS product and integrate it into the system. Depending on how complex and critical the need is, the systems engineering team is usually involved in establishing the technical outsource contract requirements, technical selection criteria, technical monitoring and control processes, and technical acceptance requirements.
- **IT services**: IT services provide capabilities that can enable an enterprise, application, or Web service solution. IT services can be provided by an outsourced service provider. In many cases, the user interface for these Web services is as simple as a Web browser. Depending on how complex and critical the needs are, the systems engineering team can be involved in establishing the technical outsourcing contract requirements, technical selection criteria, and technical acceptance process.

References

Works Cited

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense (DoD).

DoD. 2011. *Systems Engineering Plan (SEP) Outline.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

DoD. 2006. *Guide for Integrating Systems Engineering into DoD Acquisition Contracts.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

DoD. 2001. *Systems Engineering Fundamentals.* Washington, DC, USA: Defense Acquisition University Press/US Department of Defense.

Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense (DoD).

DoD. 2011. *Systems Engineering Plan (SEP) Outline.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

DoD. 2006. *Guide for Integrating Systems Engineering into DoD Acquisition Contracts.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

Additional References

MITRE. 2011. "Acquisition Systems Engineering." *Systems Engineering Guide.* Accessed March 9, 2012. Available: http://www.mitre.org/work/systems_engineering/guide/acquisition_systems_engineering/.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Knowledge Area: Systems Engineering and Industrial Engineering

Systems Engineering and Industrial Engineering

Industrial Engineering is concerned with the design, improvement and installation of integrated systems of people, materials, information, equipment and energy. It draws upon specialized knowledge and skill in the mathematical, physical, and social sciences together with the principles and methods of engineering analysis and design, to specify, predict, and evaluate the results to be obtained from such systems. (IIE 1992)

Industrial engineering (IE) encompasses several aspects of systems engineering (SE) (i.e., production planning and analysis, continuous process improvement, etc.) and also many elements of the engineered systems domain (production control, supply chain management, operations planning and preparation, operations management, etc.), as depicted in Figure 3 of the article Scope and Context of the SEBoK.

This knowledge area covers the overarching aspects of industrial engineering and describes the synergies between IE and SE.

Overview of Industrial Engineering

Industrial engineers are trained to design and analyze the components of which man-machine systems are composed. They bring together individual elements that are designed via other engineering disciplines and properly synergize these subsystems together with the people components for a completely integrated man-machine system. Industrial engineers are focused on the improvement of any system that is being designed or evaluated. They make individual human tasks more productive and efficient by optimizing flow, eliminating unnecessary motions, utilizing alternate materials to improve manufacturing, improving the flow of product through processes, and optimizing the configuration of work spaces. Fundamentally, the industrial engineer is charged with reducing costs and increasing profitability through ensuring the efficient use of human, material, physical, and/or financial resources (Salvendy 2001).

A systems engineer leverages industrial engineering knowledge to provide:

- production planning and analysis
- systems integration
- lifecycle planning and estimating
- change analysis and management
- continuous process improvement
- quality assurance
- business case analysis / return on investment
- engineering management
- systems integration

Industrial engineers complement systems engineers with knowledge in:

- supply chain management
- budgeting and economic analysis
- production line preparation

- production
- production control
- testing
- staffing, organizing, directing
- cost, schedule, and performance monitoring
- risk monitoring and control
- operations planning and preparation
- operations management

Industrial Engineering Body of Knowledge

The current overview of the industrial engineering body of knowledge is provided in the *Handbook of Industrial Engineering* (Salvendy 2001) and *Maynard's Industrial Engineering Handbook* (Zandin 2001). The Institute of Industrial Engineers (IIE 1992) is currently in the process of developing a specific industrial engineering body of knowledge. Additionally, industrial engineering terminology defines specific terms related to the industrial engineering profession. Definitions used in this section are from this reference. Turner et al. (1992) provide an overview of industrial and systems engineering.

The elements of IE include the following:

Operations Engineering

Operations engineering involves the management and control aspects of IE and works to ensure that all the necessary requirements are in place to effectively execute a business. Key areas of knowledge in this field include: product and process life cycles, forecasting, project scheduling, production scheduling, inventory management, capacity management, supply chain, distribution, and logistics. Concepts such as materials requirements planning and enterprise resource planning find their roots in this domain.

Operations Research

Operations research is the organized and systematic analysis of complex situations, such as if there is a spike in the activities of organizations of people and resources. The analysis makes use of certain specific disciplinary methods, such as probability, statistics, mathematical programming, and queuing theory. The purpose of operations research is to provide a more complete and explicit understanding of complex situations, to promote optimal performance utilizing the all the resources available. Models are developed that describe deterministic and probabilistic systems and these models are employed to aid the decision maker. Knowledge areas in operations research include linear programming, network optimization, dynamic programming, integer programming, nonlinear programming, metaheuristics, decision analysis and game theory, queuing systems, and simulation. Classic applications include the transportation problem and the assignment problem.

Production Engineering / Work Design

Production engineering is the design of a production or manufacturing process for the efficient and effective creation of a product. Included in this knowledge area is classic tool and fixture design, selection of machines to produce product, and machine design. Closely related to production engineering, work design involves such activities as process, procedural and work area design, which are geared toward supporting the efficient creation of goods and services. Knowledge in work simplification and work measurement are crucial to work design. These elements form a key foundation, along with other knowledge areas in IE, for lean principles.

Facilities Engineering and Energy Management

Facilities engineering involves attempting to achieve the optimal organization in factories, buildings, and offices. In addition to addressing the aspects of the layout inside a facility, individuals in this field also possess knowledge of material and equipment handling as well as storage and warehousing. This area also involves the optimal placement and sizing of facilities according to the activities they are required to contain. An understanding of code compliance and use of standards is incorporated. The energy management aspect of this area encompasses atmospheric systems and lighting and electrical systems. Through the development of responsible management of resources in the energy management domain, industrial engineers have established a basis in sustainability.

Ergonomics

Ergonomics is the application of knowledge in the life sciences, physical sciences, social sciences, and engineering that studies the interactions between the human and the total working environment, such as atmosphere, heat, light and sound, as well as the interactions of all tools and equipment in the workplace. Ergonomics is sometimes referred to as *human factors*. Individuals in this field have a specialized knowledge in areas such as: anthropometric principles, standing/sitting, repetitive task analysis, work capacity and fatigue, vision and lighting, hearing, sound, noise, vibration, human information processing, displays and controls, and human-machine interaction. Members in this field also consider the organizational and social aspects of a project.

Engineering Economic Analysis

Engineering economic analysis concerns techniques and methods that estimate output and evaluate the worth of commodities and services relative to their costs. Engineering economic analysis is used to evaluate system affordability. Fundamental to this knowledge area are value and utility, classification of cost, time value of money and depreciation. These are used to perform cash flow analysis, financial decision making, replacement analysis, break-even and minimum cost analysis, accounting and cost accounting. Additionally, this area involves decision making involving risk and uncertainty and estimating economic elements. Economic analysis also addresses any tax implications.

Quality and Reliability

Quality is the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs. Reliability is the ability of an item to perform a required function under stated conditions for a stated period of time. The understanding of probability and statistics form a key foundation to these concepts. Knowledge areas in quality and reliability include: quality concepts, control charts, lot acceptance sampling, rectifying inspection and auditing, design of experiments, and maintainability. Six sigma has its roots in the quality domain; however, its applicability has grown to encompass a total business management strategy.

Engineering Management

Engineering management refers to the systematic organization, allocation, and application of economic and human resources in conjunction with engineering and business practices. Knowledge areas include: organization, people, teamwork, customer focus, shared knowledge systems, business processes, resource responsibility, and external influences.

Supply Chain Management

Supply chain management deals with the management of the input of goods and services from outside sources that are required for a business to produce its own goods and services. Information is also included as a form of input. Knowledge areas include: building competitive operations, planning and logistics, managing customer and supplier relationships, and leveraging information technology to enable the supply chain.

References

Works Cited

IIE. 1992. *Industrial Engineering Terminology*, revised ddition. Norwood, GA, USA: Institute of Industrial Engineers (IIE). Accessed March 7, 2012. Available: http://www.iienet2.org/Details.aspx?id=645.

Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Turner, W.C., J.H. Mize, K.E. Case, and J.W. Nazemtz. 1992. *Introduction To Industrial And Systems Engineering*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall.

Zandin, K.B. (ed.) 2001. *Maynard's Industrial Engineering Handbook*, 5th ed. New York, NY, USA: McGraw-Hill.

Primary References

IIE. 1992. *Industrial Engineering Terminology*, revised ddition. Norwood, GA, USA: Institute of Industrial Engineers (IIE). Accessed March 7, 2012. Available: http://www.iienet2.org/Details.aspx?id=645.

Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Zandin, K.B. (ed.) 2001. *Maynard's Industrial Engineering Handbook,* 5th ed. New York, NY, USA: McGraw-Hill.

Additional References

Operations Engineering

Hopp, W., and M. Spearman. 2001. *Factory Physics,* 3rd ed., New York, NY, USA: McGraw-Hill.

Heizer, J., and B. Render. 2001. *Operations Management,* 6th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Mantel, S., J. Meredith, S. Shafer, and M. Sutton. 2008. *Project Management in Practice.* New York, NY, USA: John Wiley & Sons.

Operations Research

Banks, J., J. Carson, B. Nelson, and D. Nicol. 2005. *Discrete-Event System Simulation,* 4th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Hillier, F., and G. Lieberman. 2010. *Introduction to Operations Research,* 9th ed. New York, NY, USA: McGraw Hill.

Kelton, W. David, R. Sadowski, and D. Sturrock. 2006. *Simulation with Arena,* 4th ed. New York, NY, USA: McGraw-Hill.

Law, A. 2007. *Simulation Modelling and Analysis,* 4th ed. New York, NY, USA: McGraw-Hill.

Winston, W. and J. Goldberg. 2004. *Operations Research Applications & Algorithms,* Independence, KY, USA: Thomson Brooks/Cole.

Production Engineering / Work Design

Freivalds, A. 2009. *Niebel's Methods, Standards, and Work Design,* 12th ed. New York, NY, USA: McGraw-Hill.

Groover, M. 2007 *Work Systems: The Methods, Measurement, and Management of Work,* Upper Saddle River, NJ, USA: Pearson-Prentice Hall.

Grover, M. 2007. *Fundamentals of Modern Manufacturing,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Konz, S., and S. Johnson, 2008. *Work Design: Occupational Ergonomics,* 7th ed. Scottsdale, AZ, USA: Holcomb Hathaway.

Meyers, F., and J. Stewart, 2001 *Motion and Time Study for Lean Manufacturing,* 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall.

Facilities Engineering and Energy Management

Garcia-Diaz, A., and J. MacGregor Smith. 2008. *Facilities Planning and Design,* Upper Saddle River, NJ, USA: Pearson-Prentice Hall.

Tompkins, J., J. White, Y. Bozer, and J. Tanchoco. 2003. *Facilities Planning,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Ergonomics

Chaffin, D., and G. Andersson. 1991. *Occupational Biomechanics.* New York, NY, USA: John Wiley & Sons.

Wickens, C., S. Gordon, and Y. Liu. 2004. *An Introduction to Human factors Engineering.* Upper Saddle River, NJ, USA: Pearson-Prentice Hall.

Engineering Economic Analysis

Blank, L.T., and A.J. Tarquin. 2011. *Engineering Economy,* 7th ed. New York, NY, USA: McGraw-Hill.

Newnan, D., T. Eschenbach, and J. Lavelle. 2011. *Engineering Economic Analysis,* 11th ed. New York, NY, USA: Oxford University Press.

Parl, C. 2007. *Fundamentals of Engineering Economics.* Upper Saddle River, NJ, USA: Prentice Hall.

Thuesen, G., and W. Fabrycky. 2001. *Engineering Economy,* 9th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Quality & Reliability

Ebeling, C.E. 2005. *An Introduction to Reliability and Maintainability Engineering.* Long Grove, IL, USA: Waveland Press, Inc.

Hawkins, D., and D. Olwell. 1998. *Cumulative Sum Chars and Charting for Quality Improvement.* New York, NY, USA: Springer.

Kiemele, M., S. Schmidt, and R. Berdine. 1999. *Basic Statistics: Tools for Continuous Improvement*, 4th ed. Colorado Springs, CO, USA: Air Academy Press.

Montgomery, D., and G. Runger. 2007. *Applied Statistics and Probability for Engineers*, 4th ed. Hoboken, NJ, USA: John Wiley & Sons.

Montgomery, D. 2013. *Design & Analysis of Experiments,* 8th ed. Hoboken, NJ, USA: John Wiley & Sons.

Montgomery, D. 2009. *Introduction to Statistical Quality Control*, 6th ed. Hoboken, NJ, USA: John Wiley & Sons.

Quality Staff. 2006. *Data Quality Assessment: Statistical Methods for Practitioners.* Washington, DC, USA: Environmental Protection Agency (EPA).

Engineering Management

Gido, J., and J. Clements. 2009. *Successful Project Management.* Cincinnati, OH, USA: South Western.

Kersner, H. 2009. *A Systems Approach to Planning, Scheduling, and Controlling,* 10th ed. New York, NY, USA: John Wiley & Sons.

Supply Chain Management

Jacobs, F., and R. Chase. 2010. *Operations and Supply Chain Management.* New York, NY, USA: McGraw-Hill.

Mentzer, J. 2004. *Fundamentals of Supply Chain Management: Twelve Drivers of Competitive Advantage.* Thousand Oaks, CA, USA: Sage.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Systems Engineering and Specialty Engineering

Specialty engineering disciplines support product, service and enterprise development by applying crosscutting knowledge to system design decisions, balancing total system performance and affordability. This knowledge area presents several of the supporting engineering disciplines with a focus on the systems engineer.

To help develop a consistent picture of how SE is related to these specialty disciplines the SEBoK uses a standard set of headings: Description, Discipline Management, Discipline Relationships, Personnel Considerations, Metrics, Models, Tools, and References. Please note that not all of these sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Reliability, Availability, and Maintainability
- Human Systems Integration
- Security Engineering
- Electromagnetic Interference/Electromagnetic Compatibility
- System Resilience
- Manufacturability and Producibility
- Affordability
- Environmental Engineering
- Logistics Engineering

Specialty Requirements

The systems engineering team must ensure that specialty requirements are properly reviewed with regard to their impact on life cycle costs, development schedule, technical performance, and operational utility. For example, security requirements can impact operator workstations, electromagnetic interference requirements can impact the signal in the interfaces between subsystems, and mass-volume requirements may preclude the use of certain materials to reduce subsystem weight.

Engineering specialists audit the evolving design and resulting configuration items to ensure that the overall system performance also satisfies the specialty requirements. Including appropriate specialty engineers within each systems

engineering team assures that all system requirements are identified and balanced throughout the development cycle.

Integration of Specialty Engineering

Integration of engineering specialties into a project or program is, or should be, a major objective of systems engineering management. With properly implemented procedures, the rigor of the systems engineering process ensures participation of the specialty disciplines at key points in the technical decision making process. Special emphasis on integration is mandatory because a given design could in fact be accomplished without consideration of these “specialty” disciplines, leading to the possibility of system ineffectiveness or failure when an unexamined situation occurs in the operational environment.

For example, human factors considerations can contribute to reduced workloads and therefore lower error rates by operators in aircraft cockpits, at air-traffic consoles, or nuclear reactor stations. Similarly, mean-time-to-repair features can significantly increase overall system availability in challenging physical environments, such as mid-ocean or outer space. Specialty engineering requirements are often manifest as constraints on the overall system design space. The role of system engineering is to balance these constraints with other functionality in order to harmonize total system performance. The end goal is to produce a system that provides utility and effectiveness to the customer at an affordable price.

As depicted in Figure 1, systems engineering plays a leadership role in integrating traditional disciplines, specialty disciplines, and unique system product demands to define the system design. Relationships for this integration process are represented as interactions among three filters.

The first filter is a conceptual analysis that leverages traditional design consideration (structural, electronics, aerodynamics, mechanical, thermodynamics, and other). The second filter evaluates the conceptual approach using specialty disciplines, such as safety, affordability, quality assurance, human factors, reliability and maintainability, producibility, packaging, test, logistics, and others, to further requirements development. Design alternatives that pass through these two processes go through a third filter that incorporates facility design, equipment design, procedural data, computer programs, and personnel to develop the final requirements for design selection and further detailed development.

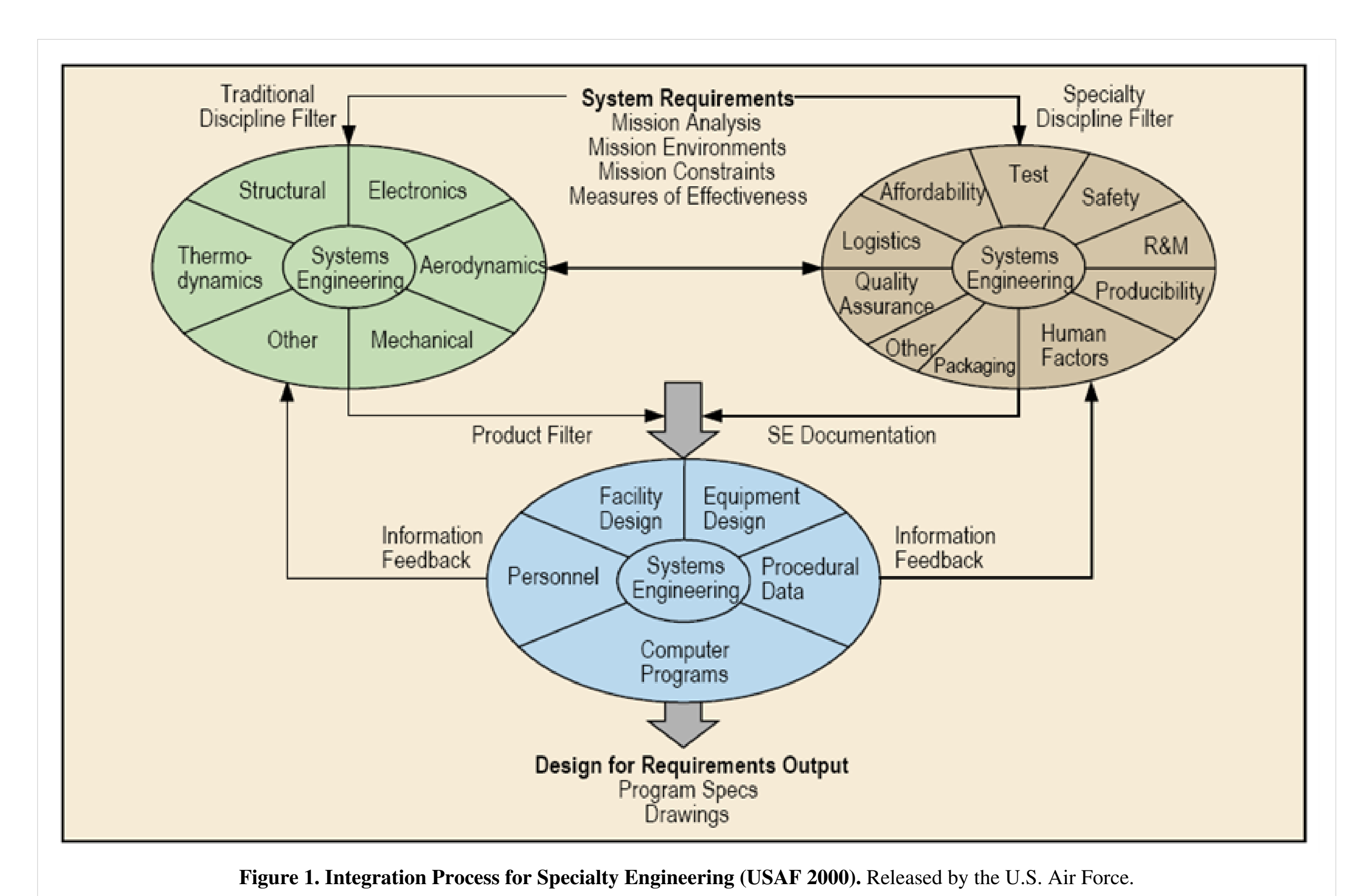


Figure 1. Integration Process for Specialty Engineering (USAF 2000). Released by the U.S. Air Force.

References

Works Cited

USAF. 2000. *Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems Command and Control Systems Management Information Systems*, version 3.0. Hill AFB: Department of the Air Force Software Technology Support Center. May 2000. Accessed on September 11, 2011. Available at http://www.stsc.hill.af.mil/resources/tech%5Fdocs/.

Primary References

USAF. 2000. *Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems Command and Control Systems Management Information Systems*, version 3.0. Hill AFB: Department of the Air Force Software Technology Support Center. May 2000. Accessed on September 11, 2011. Available at http://www.stsc.hill.af.mil/resources/tech%5Fdocs/.

Additional References

None.

< Previous Article | Parent Article | Next Article >

Reliability, Availability, and Maintainability

Reliability, availability, and maintainability (RAM) are three system attributes that are of tremendous interest to systems engineers, logisticians, and users. Collectively, they affect economic life-cycle costs of a system and its utility.

Overview

Reliability, maintainability, and availability (RAM) are three system attributes that are of great interest to systems engineers, logisticians, and users. Collectively, they affect both the utility and the life-cycle costs of a product or system. The origins of contemporary reliability engineering can be traced to World War II. The discipline's first concerns were electronic and mechanical components (Ebeling, 2010). However, current trends point to a dramatic rise in the number of industrial, military, and consumer products with integrated computing functions Because of the rapidly increasing integration of computers into products and systems used by consumers, industry, governments, and the military, reliability must consider both hardware, and software.

Maintainability models present some interesting challenges. The time to repair an item is the sum of the time required for evacuation, diagnosis, assembly of resources (parts, bays, tool, and mechanics), repair, inspection, and return. Administrative delay (such as holidays) can also affect repair times. Often these sub-processes have a minimum time to complete that is not zero, resulting in the distribution used to model maintainability having a threshold parameter.

A threshold parameter is defined as the minimum probable time to repair. Estimation of maintainability can be further complicated by queuing effects, resulting in times to repair that are not independent. This dependency frequently makes analytical solution of problems involving maintainability intractable and promotes the use of simulation to support analysis.

System Description

This section sets forth basic definitions, briefly describes probability distributions, and then discusses the role of RAM engineering during system development and operation. The final subsection lists the more common reliability test methods that span development and operation.

Basic Definitions

Reliability

defined as the probability of a system or system element performing its intended function under stated conditions without failure for a given period of time (ASQ 2011). A precise definition must include a detailed description of the function, the environment, the time scale, and what constitutes a failure. Each can be surprisingly difficult to define as precisely as one might wish.

Maintainability

defined as the probability that a system or system element can be repaired in a defined environment within a specified period of time. Increased maintainability implies shorter repair times (ASQ 2011).

Availability

probability that a repairable system or system element is operational at a given point in time under a given set of environmental conditions. Availability depends on reliability and maintainability and is discussed in detail later in this topic (ASQ 2011).

A failure is the event(s), or inoperable state, in which any item or part of an item does not, or would not, perform as specified (GEIA 2008). The failure mechanism is the physical, chemical, electrical, thermal, or other process that results in failure (GEIA 2008). In computerized system, a software defect or fault can be the cause of a failure (Laprie 1992) and the failure may have been preceded by an error which was internal to the item. The failure mode is the way or the consequence of the mechanism through which an item fails (GEIA 2008, Laprie 1992.). The severity of the failure mode is the magnitude of its impact (Laprie, 1992).

Probability Distributions used in Reliability Analysis

Reliability can be thought of as the probability of the survival of a component until time t. Its complement is the probability of failure before or at time t. If we define a random variable T as the time te to failure, then

where R(t) is the reliability and F(t) is the failure probability, The failure probability is the cumulative distribution function (CDF) of a mathematical probability distribution. Continuous distributions used for this purpose include exponential, Weibull, log-normal, and generalized gamma. Discrete distributions such as the Bernoulli, Binomial, and Poisson are used for calculating the expected number of failures or for single probabilities of success

The same continuous distributions used for reliability can also be used for maintainability although the interpretation is different (i.e., probability that a failed component is restored to service prior to time t). However, predictions of maintainability may have to account for processes such as administrative delays, travel time, sparing, and staffing and can therefore most complex.

The probability distributions used in reliability and maintainability estimation are referred to as models because they only provide estimates of the true failure and restoration of the items under evaluation. Ideally, the values of the parameters used in these models would be estimated from life testing or operating experience. However, performing such tests or collecting credible operating data once items are fielded can be costly. Therefore, approximations use of data from “similar systems”, “engineering judgment”, other methods are sometimes used. As a result, that estimates based on limited data may be very imprecise. Testing methods to gather such data are discussed below.

RAM Considerations during Systems Development

RAM are inherent product or system attributes that should be considered throughout the development lifecycle. Reliability standards, textbook authors, and others have proposed multiple development process models (O’Connor 2014, Kapur 2014, Ebeling 2010, DoD 2005). The discussion in this section relies on a standard developed by a joint effort by the Electronic Industry Association and the U.S. Government and adopted by the U.S. Department of Defense (GEIA 2008) that defines 4 processes: understanding user requirements and constraints, design for reliability, production for reliability, and monitoring during operation and use (discussed in the next section).

Understanding User Requirements and Constraints

Understanding user requirements involves eliciting information about functional requirements, constraints (e.g., mass, power consumption, spatial footprint, life cycle cost), and needs that correspondent to RAM requirements. From these emerge system requirements that should include specifications for reliability, maintainability, and availability, and each should be conditioned on the projected operating environments. RAM requirements definition is as challenging but as essential to development success as is the definition of general functional requirements.

Design for Reliability

System designs based on user requirements and system design alternatives can then be formulated and evaluated.. Reliability engineering during this phase seeks to increase system robustness through measures such as redundancy, diversity, built-in test, advanced diagnostics, and modularity to enable rapid physical replacement. In addition, it may be possible to reduce failure rates through measures such as use of higher strength materials, increasing the quality components, moderating extreme environmental conditions, or shortened maintenance, inspection, or overhaul intervals. Design analyses may include mechanical stress, corrosion, and radiation analyses for mechanical components, thermal analyses for mechanical and electrical components, and Electromagnetic Interference (EMI) analyses or measurements for electrical components and subsystems.

In most computer based systems, hardware mean time between failures are hundreds of thousands of hours so that most system design measures will be to increase system reliability are focused on software. The most obvious way to improve software reliability is by improving its quality through more disciplined development efforts and test. Methods for doing so are in the scope of software engineering but not in the scope of this section. However, reliability and availability can also be increased through architectural redundancy, independence, and diversity. Redundancy must be accompanied by measures to ensure data consistency, and managed failure detection and switchover. Within the software architecture, measures such as watchdog timers, flow control, data integrity checks (e.g., hashing or cyclic redundancy checks), input and output validity checking, retries, and restarts can increase reliability and failure detection coverage (Shooman, 2002).

System RAM characteristics should be continuously evaluated as the design progresses. Where failure rates are not known (as is often the case for unique or custom developed components, assemblies, or software), developmental testing may be undertaken assess the reliability of custom-developed components. Evaluations based on quantitative analyses assess the numerical reliability and availability of the system and are usually based on reliability block diagrams, fault trees, Markov models, and Petri nets (O'Connor, 2011). Markov models and Petri nets are of particular value for computer-based systems that use redundancy. Evaluations based on qualitative analyses assess vulnerability to single points of failure, failure containment, recovery, and maintainability. The primary qualitative methods is the failure mode effects and criticality analyses (FMECA) (Kececioglu 1991). The development program Discrepancy Reporting (DR) or Failure Reporting and Corrective Action System (FRACAS) should also be used to identify failure modes which may not have been anticipated by the FMECA and to identify common problems that can be corrected through an improved design or development process.

Analyses from related disciplines during design time also affect RAM. Human factor analyses are necessary to ensure that operators and maintainers can interact with the system in a manner that minimizes failures and the restoration times when they do occur. There is also a strong link between RAM and cybersecurity in computer based systems. On the one hand defensive measures reduce the frequency of failures due to malicious events. On the other, devices such as firewalls, policy enforcement devices, and access/authentication serves (also known as "directory servers") can also become single points of failure or performance bottlenecks that reduce system reliability and availability.

Production for Reliability

Many production issues associated with RAM are related to quality. The most important of these are ensuring repeatability and uniformity of production processes and complete unambiguous specifications for items from the supply chain. Other are related to design for manufacturability, storage, and transportation (Kapur, 2014; Eberlin 2010). Large software intensive systems information systems are affected by issues related to configuration management, integration testing, and installation testing. Testing and recording of failures in the problem reporting and corrective action systems (PRACAS) or the FRACAS capture data on failures and improvements to correct failures. Depending on organizational considerations, this may be the same or a separate system as used during the design.

Monitoring During Operation and Use

After systems are fielded, their reliability and availability to assess whether system or product has met its RAM objectives, to identify unexpected failure modes, to record fixes, to assess the utilization of maintenance resources, and to assess the operating environment. The FRACAS or a maintenance management database may be used for this purpose. In order to assess RAM, it is necessary to maintain an accurate record not only of failures but also of operating time and the duration of outages. Systems that report only on repair actions and outage incidents may not be sufficient for this purpose.

An organization should have an integrated data system that allows reliability data to be considered with logistical data, such as parts, personnel, tools, bays, transportation and evacuation, queues, and costs, allowing a total awareness of the interplay of logistical and RAM issues. These issues in turn must be integrated with management and operational systems to allow the organization to reap the benefits that can occur from complete situational awareness with respect to RAM.

Reliability and Maintainability Testing

Reliability Testing can be performed at the component, subsystem, and system level throughout the product or system lifecycle. Examples of hardware related categories of reliability testing include (Ebeling, 2010, O'Connor 2014)

- Reliability Life Tests: Reliability Life Tests are used to empirically assess the time to failure for non-repairable products and systems and the times between failure for repairable or restorable systems. Termination criteria for such tests can be based on a planned duration or planned number of failures. Methods to account for “censoring” of the failures or the surviving units enable a more accurate estimate of reliability.
- Accelerated Life Tests: Accelerated life testing is performed by subjecting the items under test (usually electronic parts) by increasing the temperature to well above the expecting operating temperature and extrapolating results using an Arhenius relation.
- Highly Accelerated Life Testing/Highly Accelerated Stress Testing (HALT/HASS) subjects units under test (components or subassemblies) to extreme temperature and vibration tests with the objective of identifying failure modes, margins, and design weaknesses.
- Parts Screening: Parts screening is not really a test but a procedure to operate components for a duration beyond the “infant mortality” period during which less durable items fail and the more durable parts that remain are then assembled into the final product or system.

Examples of system level testing (including both hardware and software) are (O'Connor 2014, Ebeling 2010)

Stability tests: Stability tests are life tests for integrated hardware and software systems. The goal of such testing is to determine the integrated system failure rate and assess operational suitability. Test conditions must include accurate simulation of the operating environment (including workload) and a means of identifying and recording failures.

- Reliability Growth Tests: Reliability Growth Testing is part of a reliability growth program in which items are tested throughout the development and early production cycle with the intent of assessing reliability increases due to improvements in the manufacturing process (for hardware) or software quality (for software)
- Failure/recovery tests: Such testing assesses the fault tolerance of a system by measuring probability of switchover for redundant systems. Failures are simulated and the ability of the hardware and software to detect the condition and reconfigure the system to remain operational are tested.
- Maintainability Tests: Such testing assess the system diagnostics capabilities, physical accessibility, and maintainer training by simulating hardware or software failures that require maintainer action for restoration.

Because of its potential impact on cost and schedule, reliability testing should be coordinated with the overall system engineering effort. Test planning considerations include the number of test units, duration of the tests, environmental conditions, and the means of detecting failures.

Data Issues

True RAM models for a system are generally never known. Data on a given system is assumed or collected, used to select a distribution for a model, and then used to fit the parameters of the distribution. This process differs significantly from the one usually taught in an introductory statistics course.

First, the normal distribution is seldom used as a life distribution, since it is defined for all negative times. Second, and more importantly, reliability data is different from classic experimental data. Reliability data is often censored, biased, observational, and missing information about covariates such as environmental conditions. Data from testing is often expensive, resulting in small sample sizes. These problems with reliability data require sophisticated strategies and processes to mitigate them.

One consequence of these issues is that estimates based on limited data can be very imprecise.

Discipline Management

In most large programs, RAM experts report to the system engineering organization. At project or product conception, top level goals are defined for RAM based on operational needs, lifecycle cost projections, and warranty cost estimates. These lead to RAM derived requirements and allocations that are approved and managed by the system engineering requirements management function. RAM testing is coordinated with other product or system testing through the testing organization, and test failures are evaluated by the RAM function through joint meetings such as a Failure Review Board. In some cases, the RAM function may recommend design or development process changes as a result of evaluation of test results or software discrepancy reports, and these proposals must be adjudicated by the system engineering organization, or in some cases, the acquiring customer if cost increases are involved.

Post-Production Management Systems

Once a system is fielded, its reliability and availability should be tracked. Doing so allows the producer / owner to verify that the design has met its RAM objectives, to identify unexpected failure modes, to record fixes, to assess the utilization of maintenance resources, and to assess the operating environment.

One such tracking system is generically known as a FRACAS system (Failure Reporting and Corrective Action System). Such a system captures data on failures and improvements to correct failures. This database is separate from a warranty data base, which is typically run by the financial function of an organization and tracks costs only.

A FRACAS for an organization is a system, and itself should be designed following systems engineering principles. In particular, a FRACAS system supports later analyses, and those analyses impose data requirements. Unfortunately, the lack of careful consideration of the backward flow from decision to analysis to model to required data too often leads to inadequate data collection systems and missing essential information. Proper prior planning

prevents this poor performance.

Of particular importance is a plan to track data on units that have not failed. Units whose precise times of failure are unknown are referred to as censored units. Inexperienced analysts frequently do not know how to analyze censored data, and they omit the censored units as a result. This can bias an analysis.

An organization should have an integrated data system that allows reliability data to be considered with logistical data, such as parts, personnel, tools, bays, transportation and evacuation, queues, and costs, allowing a total awareness of the interplay of logistical and RAM issues. These issues in turn must be integrated with management and operational systems to allow the organization to reap the benefits that can occur from complete situational awareness with respect to RAM.

Discipline Relationships

Interactions

- RAM interacts with nearly all aspects of the system development effort. Specific dependencies and interactions include:
- Systems Engineering: RAM interacts with the system engineering as described in the previous section.
- Product Management (Life Cycle Cost and Warranty): RAM interacts with the product or system lifecycle cost and warranty management organizations by assisting in the calculation of expected repair rates, downtimes, and warranty costs. RAM may work with those organizations to perform tradeoff analyses to determine the most cost efficient solution and to price service contracts.
- Quality Assurance: RAM may also interact with the procurement and quality assurance organizations with respect to selection and evaluation of materials, components, and subsystems.

Dependencies

- Systems Safety: RAM and system safety engineers have many common concerns with respect to managing the failure behavior of a system (i..e., single points of failure and failure propagation). RAM and safety engineers use similar analysis techniques, with safety being concerned about failures affecting life or unique property and RAM being concerned with those failures as well as lower severity events that disrupt operations. RAM and safety are both concerned with failures occurring during development and test – FRACAS is the primary methodology used for RAM; hazard tracking is the methodology used for system safety.
- Cybersecurity: In systems or products integrating computers and software, cybersecurity and RAM engineers have common concerns relating to the availability of cyberdefenses and system event monitoring. However, there are also tradeoffs with respect to access control, boundary devices, and authentication where security device failures could impact the availability of the product or system to users.
- Software and Hardware Engineering: Design and RAM engineers have a common goal of creating dependable product and systems. RAM interacts with the software and hardware reliability functions through design analyses such as failure modes and effects analyses, reliability predictions, thermal analyses, reliability measurement, and component specific analyses. RAM may recommend design changes as a result of these analyses that may have to be adjudicated by program management, the customer, or system engineering if there are cost or schedule impacts.
- Testing: RAM interacts with the testing program during planning to assess the most efficient (or feasible) test events to perform life testing, failure/recovery testing, and stability testing as well as to coordinate requirements for reliability or stress tests. RAM also interacts with the testing organization to assess test results and analyze failures for the implications on product or system RAM.

- Logistics: RAM works with logistics in providing expected failure rates and downtime constraints in order for logistics engineers to determine staffing, sparing, and special maintenance equipment requirements.

Discipline Standards

Because of the importance of reliability, availability, and maintainability, as well as related attributes, there are hundreds of standards associated. Some are general but more are specific to domains such as automotive, aviation, electric power distribution, nuclear energy, rail transportation, software, and many others. Standards are produced by both governmental agencies and professional associations, and international standards bodies such as

- The International Electrotechnical Commission (IEC), Geneva, Switzerland and the closely associated International Standards Organization (ISO)
- The Institute of Electrical and Electronic Engineers (IEEE), New York, NY, USA
- The Society of Automotive Engineers (SAE), Warrendale, PA, USA
- Governmental Agencies – primarily in military and space systems

The following table lists selected standards from each of these agencies. Because of differences in domains and because many standards handle the same topic in slightly different ways, selection of the appropriate requires consideration of previous practices (often documented as contractual requirements), domain specific considerations, certification agency requirements, end user requirements (if different from the acquisition or producing organization), and product or system characteristics.

Table 1. Selected Reliability, Availablity Maintainability standards (SEBoK Original)

Organization	Number, Title, and Year	Domain	Comment
IEC	IEC 60812 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA), 2006	General	
IEC	IEC 61703 Mathematical expressions for reliability, availability, maintainability and maintenance, 2001	General	
IEC	IEC 62308, Equipment reliability - Reliability assessment methods, 2006	General	
IEC	IEC 62347,Guidance on system dependability specifications, 2006	General	
IEC	IEC 62278 Railway applications – Specification and demonstration of reliability, availability, maintainability and safety (RAMS), 2002	Railways	
IEEE	IEEE,Std 352-1987, IEEE Guide for General Principles of Reliability Analysis of Nuclear Power Generating Station Safety Systems, 1987	Nuclear Energy	
IEEE	IEEE Std 1044-2009, IEEE Standard Classification for Software Anomalies, 2009	Software	
IEEE	IEEE Std 1633-2008, IEEE Recommended Practice on Software Reliability, 2008	Software	
SAE	ARP 4754A. Guidelines for the Development of Civil Aircraft and Systems, 2010	Aviation	
SAE	ARP 5890, Guidelines for Preparing Reliability Assessment Plans for Electronic Engine Controls, 2011	Aviation	
SAE	J1213/2- Use of Model Verification and Validation in Product Reliability and Confidence Assessments, 2011	General	

SAE	SAE-GEIA-STD-0009,Reliability Program Standard for Systems Design, Development, and Manufacturing, 2008	General	Used by the U.S. Dept. of Defense as the primary reliability standard (replaces MIL-STD-785B)
SAE	JA 1002. Software Reliability Program Standard, 2012	Software	
U.S. Government	NASA-STD-8729.1, Planning, Developing and Managing An Effective Reliability And Maintainability (R&M) Program	Space Systems	
U.S. Government	MIL HDBK 470A Designing And Developing Maintainable Products And Systems, 1997	Defense Systems	
U.S. Government	MIL HDBK 217F (Notice 2), Reliability Prediction of Electronic Equipment, 1995	Defense Systems	Although formally titled a "Handbook" and more than 2 decades old, the values and methods constitute a de facto standard for some U.S. military acquisitions
U.S. Government	MIL-STD-1629A, Procedures for Performing a Failure Mode Effects and Criticality Analysis - Revision A, 1980		The parent of FMEA standards produced by the IEEE, SAE, ISO, and many other agencies.,After 3 decades, Still valid and still in use

Personnel Considerations

Becoming a reliability engineer requires education in probability and statistics as well as the specific engineering domain of the product or system under development or in operation. A number of universities throughout the world have departments of reliability engineering (which also address maintainability and availability) and more have research groups and courses in reliability and safety – often within the context of another discipline such as computer science, system engineering, civil engineering, mechanical engineering, or bioengineering. Because most academic engineering programs do not have a full reliability department, most engineers working in reliability have been educated in other disciplines and acquire the additional skills through additional coursework or by working with other qualified engineers. A certification in reliability engineering is available from the American Society for Quality (ASQ 2016). However, only a minority of engineers working in the discipline have this certification.

Metrics

The three basic metrics of RAM are (not surprisingly) Reliability, Maintainability, and Availability. Reliability can be characterized in terms of the parameters, mean, or any percentile of a reliability distribution. However, in most cases, the exponential distribution is used, and a single value, the mean time to failure (MTTF) for non-restorable systems, or mean time between failures (MTBF for restorable systems are used). The metric is defined as

$$\{MTTF|MTBF\} = \frac{T_{op,Tot}}{n_f ails}$$

where $T_{op}, T_{ot}$is the total operating time and nfails is the number of failures

Maintainability is often characterized in terms of the exponential distribution and the mean time to repair and be similarly calculated, i.e.,

$$MTTR = \frac{T_{down,Tot}}{n_{outages}}$$

Where $T_{down,Tot}$is the total down time and noutages is the number of outages.

As was noted above, accounting for downtime requires definitions and specificity. Down time might be counted only for corrective maintenance actions, or it may include both corrective and preventive maintenance actions. Where the lognormal rather than the exponential distribution is used, a mean down time can still be calculated, but both the log of the downtimes and the variance must be known in order to fully characterize maintainability. Availability can be calculated from the total operating time and the downtime, or in the alternative, as a function of MTBF and MTTR (Mean Time To Repair.)as

$$A = \frac{T_{op,Tot}}{T_{down,tot} + T_{op,tot}} = \frac{MTBF}{MTBF + MTTR}$$

As was the case with maintainability, availability may be qualified as to whether it includes only unplanned failures and repairs (inherent availability) or downtime due to all causes including administrative delays, staffing outages, or spares inventory deficiencies (operational availability).

Probabilistic metrics describe system performance for RAM. Quantiles, means, and modes of the distributions used to model RAM are also useful.

Availability has some additional definitions, characterizing what downtime is counted against a system. For **inherent availability**, only downtime associated with corrective maintenance counts against the system. For **achieved availability**, downtime associated with both corrective and preventive maintenance counts against a system. Finally, **operational availability** counts all sources of downtime, including logistical and administrative, against a system.

Availability can also be calculated instantaneously, averaged over an interval, or reported as an asymptotic value. **Asymptotic availability** can be calculated easily, but care must be taken to analyze whether or not a systems settles down or settles up to the asymptotic value, as well as how long it takes until the system approaches that asymptotic value.

Reliability importance measures the effect on the system reliability of a small improvement in a component's reliability. It is defined as the partial derivative of the system reliability with respect to the reliability of a component.

Criticality is the product of a component's reliability, the consequences of a component failure, and the frequency with which a component failure results in a system failure. Criticality is a guide to prioritizing reliability improvement efforts.

Many of these metrics cannot be calculated directly because the integrals involved are intractable. They are usually estimated using simulation.

Models

There are a wide range of models that estimate and predict reliability (Meeker and Escobar 1998). Simple models, such as exponential distribution, can be useful for 'back of the envelope' calculations.

System models are used to (1) combine probabilities or their surrogates, failure rates and restoration times, at the component level to find a system level probability or (2) to evaluate a system for maintainability, single points of failure, and failure propagation. The three most common are reliability block diagrams, fault trees, and failure modes and effects analyses.

There are more sophisticated probability models used for life data analysis. These are best characterized by their failure rate behavior, which is defined as the probability that a unit fails in the next small interval of time, given it has lived until the beginning of the interval, and divided by the length of the interval.

Models can be considered for a fixed environmental condition. They can also be extended to include the effect of environmental conditions on system life. Such extended models can in turn be used for accelerated life testing (ALT), where a system is deliberately and carefully overstressed to induce failures more quickly. The data is then extrapolated to usual use conditions. This is often the only way to obtain estimates of the life of highly reliable products in a reasonable amount of time (Nelson 1990).

Also useful are **degradation models**, where some characteristic of the system is associated with the propensity of the unit to fail (Nelson 1990). As that characteristic degrades, we can estimate times of failure before they occur.

The initial developmental units of a system often do not meet their RAM specifications. **Reliability growth models** allow estimation of resources (particularly testing time) necessary before a system will mature to meet those goals (Meeker and Escobar 1998).

Maintainability models describe the time necessary to return a failed repairable system to service. They are usually the sum of a set of models describing different aspects of the maintenance process (e.g., diagnosis, repair, inspection, reporting, and evacuation). These models often have threshold parameters, which are minimum times until an event can occur.

Logistical support models attempt to describe flows through a logistics system and quantify the interaction between maintenance activities and the resources available to support those activities. Queue delays, in particular, are a major source of down time for a repairable system. A logistical support model allows one to explore the trade space between resources and availability.

All these models are abstractions of reality, and so at best approximations to reality. To the extent they provide useful insights, they are still very valuable. The more complicated the model, the more data necessary to estimate it precisely. The greater the extrapolation required for a prediction, the greater the imprecision.

Extrapolation is often unavoidable, because high reliability equipment typically can have long life and the amount of time required to observe failures may exceed test times. This requires strong assumptions be made about future life (such as the absence of masked failure modes) and that these assumptions increase uncertainty about predictions. The uncertainty introduced by strong model assumptions is often not quantified and presents an unavoidable risk to the system engineer.

There are many ways to characterize the reliability of a system, including fault trees, reliability block diagrams, and failure mode effects analysis.

A **Fault Tree** (Kececioglu 1991) is a graphical representation of the failure modes of a system. It is constructed using logical gates, with AND, OR, NOT, and K of N gates predominating. Fault trees can be complete or partial; a partial fault tree focuses on a failure mode or modes of interest. They allow 'drill down' to see the dependencies of systems on nested systems and system elements. Fault trees were pioneered by Bell Labs in the 1960s.

A Failure Mode Effects Analysis is a table that lists the possible failure modes for a system, their likelihood, and the effects of the failure. A Failure Modes Effects Criticality Analysis scores the effects by the magnitude of the product of the consequence and likelihood, allowing ranking of the severity of failure modes (Kececioglu 1991).

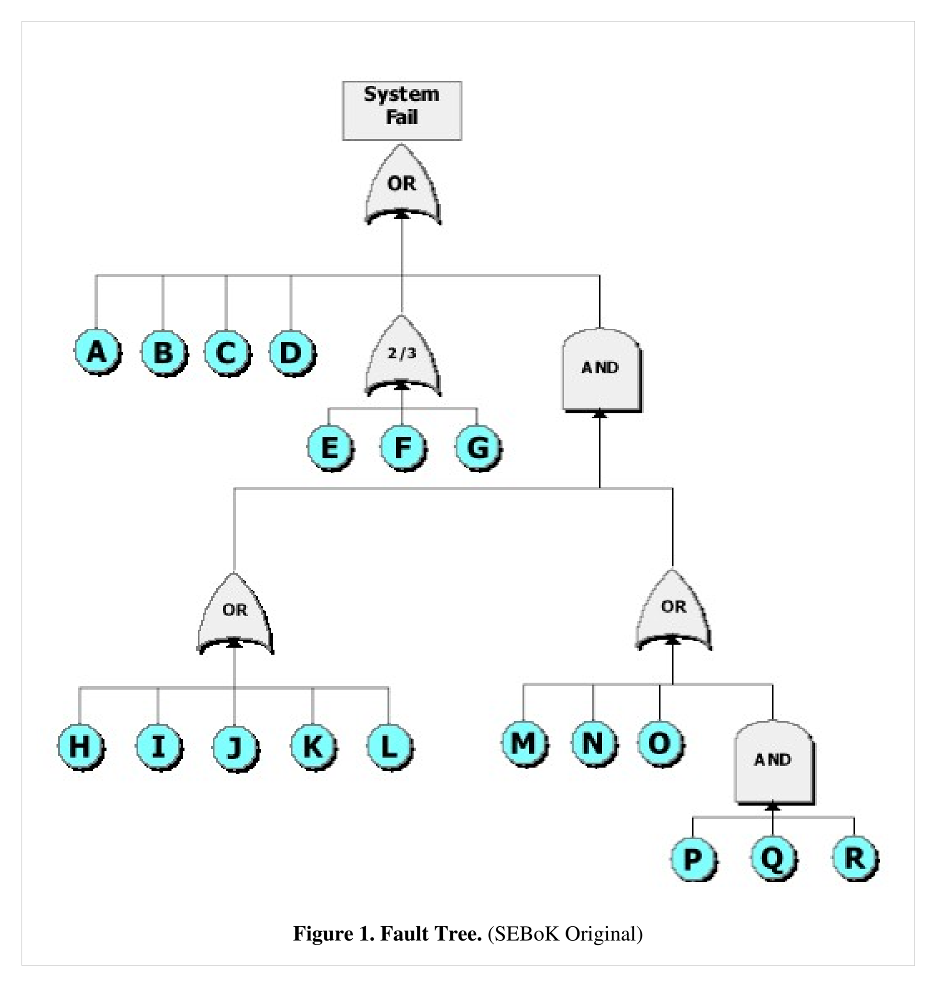


Figure 1. Fault Tree. (SEBoK Original)

A **Reliability Block Diagram** (RBD) is a graphical representation of the reliability dependence of a system on its components. It is a directed, acyclic graph. Each path through the graph represents a subset of system components. As long as the components in that path are operational, the system is operational. Component lives are usually assumed to be independent in a RBD. Simple topologies include a series system, a parallel system, a *k* of *n* system, and combinations of these.

RBDs are often nested, with one RBD serving as a component in a higher level model. These hierarchical models allow the analyst to have the appropriate resolution of detail while still permitting abstraction.

RBDs depict paths that lead to success, while fault trees depict paths that lead to failure.

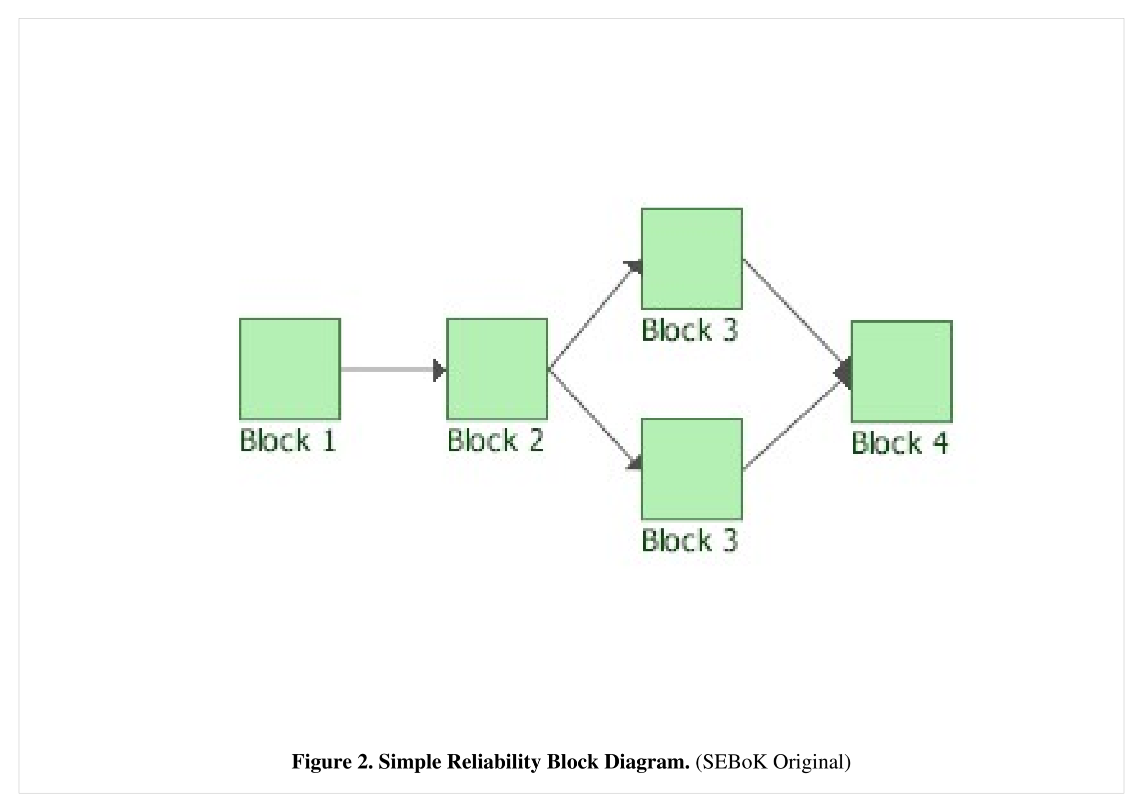


Figure 2. Simple Reliability Block Diagram. (SEBoK Original)

A **Failure Mode Effects Analysis** is a table that lists the possible failure modes for a system, their likelihood, and the effects of the failure. A **Failure Modes Effects Criticality Analysis** scores the effects by the magnitude of the product of the consequence and likelihood, allowing ranking of the severity of failure modes (Kececioglu 1991).

System models require even more data to fit them well. “Garbage in, garbage out” (GIGO) particularly applies in the case of system models.

Tools

The specialized analyses required for RAM drive the need for specialized software. While general purpose statistical languages or spreadsheets can, with sufficient effort, be used for reliability analysis, almost every serious practitioner uses specialized software.

Minitab (versions 13 and later) includes functions for life data analysis. Win Smith is a specialized package that fits reliability models to life data and can be extended for reliability growth analysis and other analyses. Relex has an extensive historical database of component reliability data and is useful for estimating system reliability in the design phase.

There is also a suite of products from ReliaSoft (2007) that is useful in specialized analyses. Weibull++ fits life models to life data. ALTA fits accelerated life models to accelerated life test data. BlockSim models system reliability, given component data.

Discipline Specific Tool Families

Reliasoft (http://www.reliasoft.com/products.htm) and PTC Windchill Product Risk and Reliability (http://www.ptc.com/product-lifecycle-management/windchill/product-risk-and-reliability) produce a comprehensive family of tools for component reliability prediction, system reliability predictions (both reliability block diagrams and fault trees), reliability growth analysis, failure modes and effects analyses, FRACAS databases, and other specialized analyses. In addition to these comprehensive tool families, there are more narrowly scoped tools. Minitab (versions 13 and later) includes functions for life data analysis.

General Purpose Statistical Analysis Software with Reliability Support

Some general purpose statistical analysis software include functions for reliability data analysis. Minitab (https://www.minitab.com/en-us/products/minitab/look-inside/) has a module for reliability and survival analysis. SuperSmith (http://www.barringer1.com/wins.htm) is a more specialized package that fits reliability models to life data and can be extended for reliability growth analysis and other analyses.

R (https://www.r-project.org/) is a widely used open source and well supported general purpose statistical language with specialized packages that can be used for fitting reliability models, Bayesian analysis, and Markov modeling.

Special Purpose Analysis Tools

Fault tree generation and analysis tools include CAFTA (http://teams.epri.com/RR/News%20Archives/CAFTAFactSheet.pdf) from the Electric Power Research Institute and OpenFTA (http://www.openfta.com/), an open source software tool originally developed by Auvation Software.

PRISM (http://www.prismmodelchecker.org/) is an open source probabilistic model checker that can be used for Markov modeling (both continuous and discrete time) as well as for more elaborate analyses of system (more specifically, “timed automata”) behaviors such as communication protocols with uncertainty.

Practical Considerations

Pitfalls

Information to be provided at a later date.

Proven Practices

Information to be provided at a later date.

Other Considerations

Information to be provided at a later date.

References

Works Cited

American Society for Quality (ASQ). 2011. *Glossary: Reliability.* Accessed on September 11, 2011. Available at http://asq.org/glossary/r.html.

American Society for Quality (ASQ) 2016, Reliability Engineering Certification – CRE, information available online at http://asq.org/cert/reliability-engineer

DoD. 2005. DOD Guide for Achieving Reliability, Availability, and Maintainability. Arlington, VA, USA: U.S. Department of Defense (DoD). Accessed on September 11, 2011. Available at:http://www.acq.osd.mil/se/docs/RAM_Guide_080305.pdf

Ebeling, Charles E., 2010. An Introduction to Reliability and Maintainability Engineering, Long Grove Illinois, U.S.A: Waveland Press.

GEIA 2008, Reliability Program Standard for Systems Design, Development, and Manufacturing, 2008, Warrendale, PA,USA: Society of Automotive Engineers (SAE), SAE-GEIA-STD-0009.

IEEE. 2008. IEEE Recommended Practice on Software Reliability. New York, NY, USA: Institute of Electrical and Electronic Engineers (IEEE). IEEE Std 1633-2008.

Kececioglu, D. 1991. *Reliability Engineering Handbook,* Volume 2. Upper Saddle River, NJ, USA: Prentice Hall.

Laprie, J.C., A. Avizienis, and B. Randell, 1992, Dependability: Basic Concepts and Terminology, Vienna: Springer-Verlag.

Nelson, Wayne. 1990. Accelerated Testing: Statistical Models, Test Plans, and Data Analysis. New York, NY, USA: Wiley and Sons.

O'Connor, D.T., and A. Kleyner, 2012, "Practical Reliability Engineering", 5th Edition. Chichester, UK: J. Wiley & Sons, Ltd.

ReliaSoft. 2007. *Failure Modes and Effects Analysis (FMEA) and Failure Modes, Effects and Criticality Analysis (FMECA).* Accessed on September 11, 2011. Available at http://www.weibull.com/basics/fmea.htm.

Shooman, Martin, 2002, Reliability of Computer Systems and Networks, New York, John Wiley & Sons

Primary References

Blischke, W.R. and D.N. Prabhakar Murthy. 2000. *Reliability Modeling, Prediction, and Optimization.* New York, NY, USA: Wiley and Sons.

Dezfuli, Homayan, Dana Kelly, Curtis Smith, Kurt vedros, and William Galyean, "Bayesian Inference for NASA Risk and Reliability Analysis", National Aeronautics and Space Administration, NASA/SP-2009-569, June, 2009, available online at http://www.hq.nasa.gov/office/codeq/doctree/SP2009569.pdf

DoD. 2005. *DOD Guide for Achieving Reliability, Availability, and Maintainability.* Arlington, VA, USA: U.S. Department of Defense (DoD). Accessed on September 11, 2011. Available at: http://www.acq.osd.mil/se/docs/RAM_Guide_080305.pdf [1]

Kececioglu, D. 1991. *Reliability Engineering Handbook,* Volume 2. Upper Saddle River, NJ, USA: Prentice Hall.

Lawless, J.F. 1982. *Statistical Models and Methods for Lifetime Data.* New York, NY, USA: Wiley and Sons.

Lyu, Michael, 1996 Software Reliability Engineering, New York, NY: IEEE-Wiley Press http://www.cse.cuhk.edu.hk/~lyu/book/reliability/index.html

Martz, H.F. and R.A. Waller. 1991. *Bayesian Reliability Analysis.* Malabar, FL, USA: Kreiger.

Meeker, W.Q. and L.A. Escobar. 1998. *Statistical Methods for Reliability Data.* New York, NY, USA: Wiley and Sons.

MIL-HDBK-189C, Department Of Defense Handbook: Reliability Growth Management (14 JUN 2011) available online from http://everyspec.com/MIL-HDBK/MIL-HDBK-0099-0199/MIL-HDBK-189C_34842

MIL-HDBK-338B, Electronic Reliability Design Handbook, 1998, U.S. Department of Defense Air Force Research Laboratory IFTB, available online at http://www.weibull.com/mil_std/mil_hdbk_338b.pdf

U.S. Naval Surface Weapons Center Carderock Division, NSWC-11, Handbook of Reliability Prediction Procedures for Mechanical Equipment, available online from http://reliabilityanalyticstoolkit.appspot.com/static/Handbook_of_Reliability_Prediction_Procedures_for Mechanical_Equipment_NSWC-11.pdf

Additional References

IEEE Recommended Practice for Collecting Data for Use in Reliability, Availability, and Maintainability Assessments of Industrial and Commercial Power Systems, IEEE Std 3006.9-2013.

NIST Sematech Engineering Statistics Handbook 2013, available online at http://www.itl.nist.gov/div898/handbook/

Olwell, D.H. 2011. "Reliability Leadership." *Proceedings of the 2001 Reliability and Maintainability M Symposium.* Philadelphia, PA, USA: IEEE. Accessed 7 March 2012 at [IEEE web site. [2]]

Reliability Analytics Toolkit, http://reliabilityanalyticstoolkit.appspot.com/ (web page containing 31 reliability and statistical analyses calculation aids), Seymour Morris, Reliability Analytics, last visited July 4, 2016

ReliaSoft. 2007. "Availability." Accessed on September 11, 2011. Available at: http://www.weibull.com/SystemRelWeb/availability.htm.

SAE. 2000a. *Aerospace Recommended Practice ARP5580: Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications.* Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

SAE. 2000b. *Surface Vehicle Recommended Practice J1739: (R) Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA), and Potential Failure Mode and Effects Analysis for Machinery (Machinery FMEA).* Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.acq.osd.mil/se/docs/RAM_Guide_080305.pdf
[2] http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=902456

Human Systems Integration

Human systems integration (HSI) is an interdisciplinary technical and management process for integrating human considerations with and across all system elements, an essential enabler to systems engineering practice. Human activity considered by HSI includes operating, maintaining, and supporting the system. HSI also considers training and training devices, as well as the infrastructure used for operations and support (DAU 2010). HSI incorporates the following domains as integration considerations: manpower, personnel, training, human factors engineering, occupational health, environment, safety, habitability, and human survivability.

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Human factors engineering is primarily concerned with designing human-machine interfaces consistent with the physical, cognitive, and sensory abilities of the user population. Human-machine interfaces include: •functional interfaces (functions and tasks, and allocation of functions to human performance or automation); •informational interfaces (information and characteristics of information that provide the human with the knowledge, understanding, and awareness of what is happening in the tactical environment and in the system); •environmental interfaces (the natural and artificial environments, environmental controls, and facility design); •co-operational interfaces (provisions for team performance, cooperation, collaboration, and communication among team members and with other personnel); •organizational interfaces (job design, management structure, command authority, and policies and regulations that impact behavior); •operational interfaces (aspects of a system that support successful operation of the system such as procedures, documentation, workloads, and job aids); •cognitive interfaces (decision rules, decision support systems, provisions for maintaining situational awareness, mental models of the tactical environment, provisions for knowledge generation, cognitive skills and attitudes, and memory aids); and •physical interfaces (hardware and software elements designed to enable and facilitate effective and safe human performance such as controls, displays, workstations, worksites, accesses, labels and markings, structures, steps and ladders, handholds, maintenance provisions, etc.) (DAU 2010).

System Description

HSI is more than human factors, human-computer interaction, or systems engineering. It is an technical and managerial set of processes that involves the consideration and integration of multiple domains. Various organizations represent the HSI domains differently as the number and names of the domains are aligned with existing organizational structures. Booher (2003) presents the seven US Army domains. The Canadian Forces have a different number of domains while the UK Ministry of Defense has another. All the technical work of the domains is present while the number and names and the domains is the same. According to the Defense Acquisition University, the HSI domains are Manpower: Manpower describes the number and mix of personnel required to carry out a task, multiple tasks, or mission in order to operate, maintain, support, and provide training for a system. Manpower factors are those variables that define manpower requirements. These variables include job tasks, operation/maintenance rates, associated workload, and operational conditions (e.g., risk of operator injury) (DAU 2010).

Environment: Environment includes the physical conditions in and around the system, as well as the operational context within which the system will be operated and supported. Environmental attributes include temperature, humidity, noise, vibration, radiation, shock, air quality, among many others. This "environment" affects the human's ability to function as a part of the system (DAU 2010).

Habitability: Habitability factors are those living and working conditions that are necessary to sustain the morale, safety, health, and comfort of the user population. They directly contribute to personnel effectiveness and mission

accomplishment and often preclude recruitment and retention problems. Examples include: lighting, space, ventilation, and sanitation; noise and temperature control (i.e., heating and air conditioning); religious, medical, and food services availability; and berthing, bathing, and personal hygiene. Habitability consists of those characteristics of systems, facilities (temporary and permanent), and services necessary to satisfy personnel needs. Habitability factors are those living and working conditions that result in levels of personnel morale, safety, health, and comfort adequate to sustain maximum personnel effectiveness, support mission performance, and avoid personnel retention problems (DAU 2010). Safety: The design features and operating characteristics of a system that serve to minimize the potential for human or machine errors or failure that cause injurious accidents (DAU, 2010). Safety also encompasses the administrative procedures and controls associated with the operations, maintenance, and storage of a system.

Human factors engineering: Human factors engineering is primarily concerned with designing human-machine interfaces consistent with the physical, cognitive, and sensory abilities of the user population. Human-machine interfaces include: •functional interfaces (functions and tasks, and allocation of functions to human performance or automation); •informational interfaces (information and characteristics of information that provide the human with the knowledge, understanding, and awareness of what is happening in the tactical environment and in the system); •environmental interfaces (the natural and artificial environments, environmental controls, and facility design); •co-operational interfaces (provisions for team performance, cooperation, collaboration, and communication among team members and with other personnel); •organizational interfaces (job design, management structure, command authority, and policies and regulations that impact behavior); •operational interfaces (aspects of a system that support successful operation of the system such as procedures, documentation, workloads, and job aids); •cognitive interfaces (decision rules, decision support systems, provisions for maintaining situational awareness, mental models of the tactical environment, provisions for knowledge generation, cognitive skills and attitudes, and memory aids); and •physical interfaces (hardware and software elements designed to enable and facilitate effective and safe human performance such as controls, displays, workstations, worksites, accesses, labels and markings, structures, steps and ladders, handholds, maintenance provisions, etc.) (DAU 2010).

Human survivability: Survivability factors consist of those system design features that reduce the risk of fratricide, detection, and the probability of being attacked, and that enable personnel to withstand man-made hostile environments without aborting the mission, objective, or suffering acute chronic illness, disability, or death. Survivability attributes are those that contribute to the survivability of manned systems (DAU 2010).

Occupational health: Occupational health factors are those system design features that serve to minimize the risk of injury, acute or chronic illness, or disability, and/or reduce job performance of personnel who operate, maintain, or support the system. Prevalent issues include noise, chemical safety, atmospheric hazards (including those associated with confined space entry and oxygen deficiency), vibration, ionizing and non-ionizing radiation, and human factors issues that can create chronic disease and discomfort such as repetitive motion diseases. Many occupational health problems, particularly noise and chemical management, overlap with environmental impacts. Human factors stresses that creating a risk of chronic disease and discomfort overlaps with occupational health considerations (DAU 2010).

Personnel: Personnel factors are those human aptitudes (i.e., cognitive, physical, and sensory capabilities), knowledge, skills, abilities, and experience levels that are needed to properly perform job tasks. Personnel factors are used to develop occupational specialties for system operators, maintainers, trainers, and support personnel (DAU 2010). The selection and assignment of personnel is critical to the success of a system, as determined by the needs set up by various work-related requirements.

Safety: The design features and operating characteristics of a system that serve to minimize the potential for human or machine errors or failure that cause injurious accidents (DAU, 2010). Safety also encompasses the administrative procedures and controls associated with the operations, maintenance, and storage of a system.

Training: Training is the learning process by which personnel individually or collectively acquire or enhance pre-determined job-relevant knowledge, skills, and abilities by developing their cognitive, physical, sensory, and

team dynamic abilities. The "training/instructional system" integrates training concepts and strategies, as well as elements of logistic support to satisfy personnel performance levels required to operate, maintain, and support the systems. It includes the "tools" used to provide learning experiences, such as computer-based interactive courseware, simulators, actual equipment (including embedded training capabilities on actual equipment), job performance aids, and Interactive Electronic Technical Manuals (DAU 2010).

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Information to be supplied at a later date.

Discipline Standards

Information to be provided at a later date.

Personnel Considerations

Personnel factors are those human aptitudes (i.e., cognitive, physical, and sensory capabilities), knowledge, skills, abilities, and experience levels that are needed to properly perform job tasks. Personnel factors are used to develop occupational specialties for system operators, maintainers, trainers, and support personnel (DAU 2010). The selection and assignment of personnel is critical to the success of a system, as determined by the needs set up by various work-related requirements.

Manpower describes the number and mix of personnel required to carry out a task, multiple tasks, or mission in order to operate, maintain, support, and provide training for a system. Manpower factors are those variables that define manpower requirements. These variables include job tasks, operation/maintenance rates, associated workload, and operational conditions (e.g., risk of operator injury) (DAU 2010).

1. **Training:** Training is the learning process by which personnel individually or collectively acquire or enhance pre-determined job-relevant knowledge, skills, and abilities by developing their cognitive, physical, sensory, and team dynamic abilities. The "training/instructional system" integrates training concepts and strategies, as well as elements of logistic support to satisfy personnel performance levels required to operate, maintain, and support the systems. It includes the "tools" used to provide learning experiences, such as computer-based interactive courseware, simulators, actual equipment (including embedded training capabilities on actual equipment), job performance aids, and Interactive Electronic Technical Manuals (DAU 2010).

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be supplied at a later date.

Practical Considerations

Information to be supplied at a later date.

Pitfalls

Information to be supplied at a later date.

Proven Practices

Information to be supplied at a later date.

Other Considerations

Information to be supplied at a later date.

References

Works Cited

Booher, H.R. (ed.). 2003. *Handbook of Human Systems Integration.* Hoboken, NJ, USA: Wiley.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

US Air Force. 2009. *Air Force Human Systems Integration Handbook.* Brooks City-Base, TX, USA: Directorate of Human Performance Integration. Available at http:/ / www. wpafb. af. mil/ shared/ media/ document/ AFD-090121-054.pdf. [1]

Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Additional References

Blanchard, B. S., and W. J. Fabrycky. 2011. *Systems Engineering and Analysis*. 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Helander, Martin, Landauer, T.K, and Prabhu, P.V. 1997. *Handbook of Human-Computer Interaction.* Amsterdam, Netherlands: Elsevier.

Pew, R.W. and A.S. Mavor. 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: National Academies Press.

Wickens, C.D., Lee, J. D, Liu, Y., and Becker, S.E. Gordon. 2004. *An Introduction to Human Factors Engineering.* Englewood Cliffs, NJ, USA: Prentice-Hall.

Woodson, W.E, Tillman, B. and Tillman, P. 1992. "Human Factors Design Handbook: Information and Guidelines for the Design of Systems, Facilities, Equipment, and Products for Human Use." 2nd Ed. New York, NY, USA: McGraw Hill.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.wpafb.af.mil/shared/media/document/AFD-090121-054.pdf

Safety Engineering

In the most general sense, safety is freedom from harm. As an engineering discipline, system safety is concerned with minimizing hazards that can result in a mishap with an expected severity and with a predicted probability. These events can occur in elements of life-critical systems as well as other system elements. MIL-STD-882E defines system safety as "the application of engineering and management principles, criteria, and techniques to achieve acceptable risk, within the constraints of operational effectiveness and suitability, time, and cost, throughout all phases of the system life cycle" (DoD 2012). MIL-STD-882E defines standard practices and methods to apply as engineering tools in the practice of system safety. These tools are applied to both hardware and software elements of the system in question."

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

System safety engineering focuses on identifying hazards, their causal factors, and predicting the resultant severity and probability. The ultimate goal of the process is to reduce or eliminate the severity and probability of the identified hazards, and to minimize risk and severity where the hazards cannot be eliminated. MIL STD 882E defines a hazard as "A real or potential condition that could lead to an unplanned event or series of events (i.e., mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment." (DoD 2012).

While Systems safety engineering attempt to minimize safety issues throughout the planning and design of systems, mishaps do occur from combinations of unlikely hazards with minimal probabilities. As a result, safety engineering is often performed in reaction to adverse events after deployment. For example, many improvements in aircraft safety come about as a result of recommendations by the National Air Traffic Safety Board based on accident investigations. Risk is defined as "A combination of the severity of the mishap and the probability that the mishap will occur" (DoD 2012, 7). Failure to identify risks to safety, and the according inability to address or "control" these risks, can result in massive costs, both human and economic (Roland and Moriarty 1990)."

System Description

Information to be supplied at a later date.

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Information to be supplied at a later date.

Discipline Standards

Information to be supplied at a later date.

Personnel Considerations

System Safety specialists are typically responsible for ensuring system safety. Air Force Instruction (AFI) provides the following guidance:

> *9.1 System safety disciplines apply engineering and management principles, criteria, and techniques throughout the life cycle of a system within the constraints of operational effectiveness, schedule, and costs.*
>
> *9.1.1. System safety is an inherent element of system design and is essential to supporting system requirements. Successful system safety efforts depend on clearly defined safety objectives and system requirements.*
>
> *9.1.2. System safety must be a planned, integrated, comprehensive effort employing both engineering and management resources.*

(USAF 1998, 91-202)

Safety personnel are responsible for the integration of system safety requirements, principles, procedures, and processes into the program and into lower system design levels to ensure a safe and effective interface. Two common mechanisms are the Safety Working Group (SWG) and the Management Safety Review Board (MSRB). The SWG enables safety personnel from all integrated product teams (IPTs) to evaluate, coordinate, and implement a safety approach that is integrated at the system level in accordance with MIL-STD-882E (DoD 2012). Increasingly, safety reviews are being recognized as an important risk management tool. The MSRB provides program level oversight and resolves safety related program issues across all IPTs. Table 1 provides additional information on safety.

Table 1. Safety Ontology. (SEBoK Original)

Ontology Element Name	Ontology Element Attributes	Relationships to Safety
Failure modes	Manner of failure	Required attribute
Severity	Consequences of failure	Required attribute
Criticality	Impact of failure	Required attribute
Hazard Identification	Identification of potential failure modes	Required to determine failure modes
Risk	Probability of a failure occurring	Required attribute
Mitigation	Measure to take corrective action	Necessary to determine criticality and severity

Table 1. indicates that achieving System safety involves a close tie between Safety Engineering and other specialty Systems Engineering disciplines such as Reliability and Maintainability Engineering.

System safety engineering focuses on identifying hazards, their causal factors, and predicting the resultant severity and probability. The ultimate goal of the process is to reduce or eliminate the severity and probability of the identified hazards, and to minimize risk and severity where the hazards cannot be eliminated. MIL STD 882E defines a hazard as "A real or potential condition that could lead to an unplanned event or series of events (i.e., mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment." (DoD 2012).

While Systems safety engineering attempt to minimize safety issues throughout the planning and design of systems, mishaps do occur from combinations of unlikely hazards with minimal probabilities. As a result, safety engineering is often performed in reaction to adverse events after deployment. For example, many improvements in aircraft safety come about as a result of recommendations by the National Air Traffic Safety Board based on accident investigations. Risk is defined as “A combination of the severity of the mishap and the probability that the mishap will occur" (DoD 2012, 7). Failure to identify risks to safety, and the according inability to address or "control" these risks, can result in massive costs, both human and economic (Roland and Moriarty 1990)."

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be supplied at a later date.

References

Works Cited

DoD. 2012. *Standard practice for System Safety.* Arlington, VA, USA: Department of Defense (DoD). MIL-STD 882E. Accessed 4 November 2014 at http://assistdoc1.dla.mil/qsDocDetails.aspx?ident_number=36027

Roland, H.E. and B. Moriarty. 1990. *System Safety Engineering and Management.* Hoboken, NJ, USA: Wiley-IEEE.

USAF. 1998. *The US Air Force Mishap Prevention Program.* Washington, DC, USA: US Air Force, Air Force Instruction (AFI).

Primary References

None.

Additional References

Bahr, N. J. 2001. "System Safety Engineering and Risk Assessment." In *International Encyclopedia of Ergonomics and Human Factors.* Vol. 3. Ed. Karwowski, Waldemar. New York, NY, USA: Taylor and Francis.

ISSS. "System Safety Hazard Analysis Report." The International System Safety Society (ISSS). DI-SAFT-80101B. http://www.system-safety.org/Documents/DI-SAFT-80101B_SSHAR.DOC.

ISSS. "Safety Assessment Report." The International System Safety Society (ISSS). DI-SAFT-80102B. http:// www.system-safety.org/Documents/DI-SAFT-80102B_SAR.DOC.

ISSS. "Engineering Change Proposal System Safety Report." The International System Safety Society (ISSS). DI-SAFT-80103B. http://www.system-safety.org/Documents/DI-SAFT-80103B_ECPSSR.DOC.

ISSS. "Waiver or Deviation System Safety Report." The International System Safety Society (ISSS). DI-SAFT-80104B. http://www.system-safety.org/Documents/DI-SAFT-80104B_WDSSR.DOC.

ISSS. "System Safety Program Progress Report." The International System Safety Society (ISSS). DI-SAFT-80105B. http://www.system-safety.org/Documents/DI-SAFT-80105B_SSPPR.DOC.

ISSS. "Health Hazard Assessment Report." The International System Safety Society (ISSS). DI-SAFT-80106B. http://www.system-safety.org/Documents/DI-SAFT-80106B_HHAR.DOC.

ISSS. "Explosive Ordnance Disposal Data." The International System Safety Society (ISSS). DI-SAFT-80931B. http://www.system-safety.org/Documents/DI-SAFT-80931B_EODD.pdf.

ISSS. "Explosive Hazard Classification Data." The International System Safety Society (ISSS). DI-SAFT-81299B. http://www.system-safety.org/Documents/DI-SAFT-81299B_EHCD.pdf.

ISSS. "System Safety Program Plan (SSPP)." The International System Safety Society (ISSS). DI-SAFT-81626. http://www.system-safety.org/Documents/DI-SAFT-81626_SSPP.pdf.

ISSS. "Mishap Risk Assessment Report." The International System Safety Society (ISSS). DI-SAFT-81300A. http:// www.system-safety.org/Documents/DI-SAFT-81300A_MRAR.DOC.

Joint Software System Safety Committee. 1999. *Software System Safety Handbook.* Accessed 7 March 2012 at http:/ /www.system-safety.org/Documents/Software_System_Safety_Handbook.pdf.

Leveson, N. 2011. Engineering a safer world: systems thinking applied to safety. Cambridge, Mass: MIT Press.

Leveson, N. G. 2012. “Complexity and Safety.” In *Complex Systems Design & Management,* ed. Omar Hammami, Daniel Krob, and Jean-Luc Voirin, 27–39. Springer Berlin Heidelberg. http:/ / dx. doi. org/ 10. 1007/ 978-3-642-25203-7_2.

NASA. 2004. *NASA Software Safety Guidebook.* Accessed 7 March 2012 at [[1]].

Roland, H. E., and Moriarty, B. 1985. *System Safety Engineering and Management.* New York, NY, USA: John Wiley.

SAE. 1996. *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* ARP 4761. Warrendale, PA, USA: Society of Automotive Engineers. Accessed 28 August 2012 at [http:/ /standards.sae.org/arp4761/ [2]].

SAE. 1996. *Certification Considerations for Highly-Integrated Or Complex Aircraft Systems.* ARP 4754. Warrendale, PA, USA: Society of Automotive Engineers. Accessed 28 August 2012 at [http://standards.sae.org/ arp4754/ [3]].

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf
[2] http://standards.sae.org/arp4761/
[3] http://standards.sae.org/arp4754/

Security Engineering

Security engineering is concerned with building systems that remain secure despite malice or error. It focuses on the tools, processes, and methods needed to design and implement complete systems that proactively and reactively mitigate vulnerabilities. Security engineering is a primary discipline used to achieve system assurance.

The term System Security Engineering (SSE) is used to denote this specialty engineering field and the US Department of Defense define it as: *"an element of system engineering that applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities"* (DODI5200.44, 12).

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Security engineering incorporates a number of cross-disciplinary skills, including cryptography, computer security, tamper-resistant hardware, applied psychology, supply chain management, and law. Security requirements differ greatly from one system to the next. System security often has many layers built on user authentication, transaction accountability, message secrecy, and fault tolerance. The challenges are protecting the right items rather than the wrong items and protecting the right items but not in the wrong way.

Security engineering is an area of increasing emphasis in the defense domain. Baldwin et al. (2012) provide a survey of the issues and a detailed reference list.

The primary objective of System Security Engineering (SSE) is to minimize or contain defense system vulnerabilities to known or postulated security threats and to ensure that developed systems protect against these threats. Engineering principles and practices are applied during all system development phases to identify and reduce these system vulnerabilities to the identified system threats.

The basic premise of SSE is recognition that an initial investment in “engineering out” security vulnerabilities and “designing-in” countermeasures is a long-term benefit and cost saving measure. Further, SSE provides a means to ensure adequate consideration of security requirements, and, when appropriate, that specific security-related designs are incorporated into the overall system design during the engineering development program. Security requirements include: physical; personnel; procedural; emission; transmission; cryptographic; communications; operations; and, computer security.

There may be some variation in the SSE process from program to program, due mainly to the level of design assurance—that is, ensuring that appropriate security controls have been implemented correctly as planned—required of the contractor. These assurance requirements are elicited early in the program (where they can be adequately planned), implemented, and verified in due course of the system development.

The System Security Engineering Management Plan (SSEMP) is a key document to develop for SSE. The SSEMP identifies the planned security tasks for the program and the organizations and individuals responsible for security

aspects of the system. The goals of the SSEMP are to ensure that pertinent security issues are raised at the appropriate points in the program, to ensure adequate precautions are taken during design, implementation, test, and fielding, and to ensure that only an acceptable level of risk is incurred when the system is released for fielding. The SSEMP forms the basis for an agreement with SSE representing the developer, the government program office, the certifier, the accreditor, and any additional organizations that have a stake in the security of the system. The SSEMP identifies the major tasks for certification & accreditation (C&A), document preparation, system evaluation, and engineering; identifies the responsible organizations for each task; and presents a schedule for the completion of those tasks.

SSE security planning and risk management planning includes task and event planning associated with establishing statements of work and detailed work plans as well as preparation and negotiation of SSE plans with project stakeholders. For each program, SSE provides the System Security Plan (SSP) or equivalent. An initial system security Concept of Operations (CONOPS) may also be developed. The SSP provides: the initial planning of the proposed SSE work scope; detailed descriptions of SSE activities performed throughout the system development life cycle; the operating conditions of the system; the security requirements; the initial SSE risk assessment (includes risks due to known system vulnerabilities and their potential impacts due to compromise and/or data loss); and, the expected verification approach and validation results.

These plans are submitted with the proposal and updated as required during engineering development. In the case where a formal C&A is contracted and implemented, these plans comply with the government's C&A process, certification responsibilities, and other agreement details, as appropriate. The C&A process is the documented agreement between the customer and contractor on the certification boundary. Upon agreement of the stakeholders, these plans guide SSE activities throughout the system development life cycle.

System Assurance

NATO AEP-67 (Edition 1), Engineering for System Assurance in NATO Programs, defines system assurance as:

…the justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle... This confidence is achieved by system assurance activities, which include a planned, systematic set of multi-disciplinary activities to achieve the acceptable measures of system assurance and manage the risk of exploitable vulnerabilities. (NATO 2010, 1)

The NATO document is organized based on the life cycle processes in ISO/IEC 15288:2008 and provides process and technology guidance to improve system assurance.

Software Assurance

Since most modern systems derive a good portion of their functionality from software, software assurance becomes a primary consideration in systems assurance. The Committee on National Security Systems (CNSS) (2010, 69) defines software assurance as a "level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle and that the software functions in the intended manner."

Goertzel, et. al (2008, 8) point out that "the reason software assurance matters is that so many business activities and critical functions—from national defense to banking to healthcare to telecommunications to aviation to control of hazardous materials—depend on the on the correct, predictable operation of software."

System Description

Robust security design explicitly rather than implicitly defines the protection goals. The Certified Information Systems Security Professional (CISSP) Common Body of Knowledge (CBK) partitions robust security into ten domains (Tipton 2006):

1. Information security governance and risk management addresses the framework, principles, policies, and standards that establish the criteria and then assess the effectiveness of information protection. Security risk management contains governance issues, organizational behavior, ethics, and security awareness training.

2. Access control is the procedures and mechanisms that enable system administrators to allow or restrict operation and content of a system. Access control policies determine what processes, resources, and operations users can invoke.

3. Cryptography can be defined as the principles and methods of disguising information to ensure its integrity, confidentiality, and authenticity during communications and while in storage. Type I devices are certified by the US National Security Agency (NSA) for classified information processing. Type 2 devices are certified by NSA for proprietary information processing. Type 3 devices are certified by NSA for general information processing. Type 4 devices are produced by industry or other nations without any formal certification.

4. Physical (environmental) security addresses the actual environment configuration, security procedures, countermeasures, and recovery strategies to protect the equipment and its location. These measures include separate processing facilities, restricted access into those facilities, and sweeps to detect eavesdropping devices.

5. Security architecture and design contains the concepts, processes, principles, and standards used to define, design, and implement secure applications, operating systems, networks, and equipment. The security architecture must integrate various levels of confidentiality, integrity, and availability to ensure effective operations and adherence to governance.

6. Business continuity and disaster recovery planning are the preparations and practices which ensure business survival given events, natural or man-made, which cause a major disruption in normal business operations. Processes and specific action plans must be selected to prudently protect business processes and to ensure timely restoration.

7. Telecommunications and network security are the transmission methods and security measures used to provide integrity, availability, and confidentiality of data during transfer over private and public communication networks.

8. Application development security involves the controls applied to application software in a centralized or distributed environment. Application software includes tools, operating systems, data warehouses, and knowledge systems.

9. Operations security is focused on providing system availability for end users while protecting data processing resources both in centralized data processing centers and in distributed client/server environments.

10. Legal, regulations, investigations, and compliance issues include the investigative measures to determine if an incident has occurred and the processes for responding to such incidents.

One response to the complexity and diversity of security needs and domains that contribute to system security is “defense in depth,” a commonly applied architecture and design approach. Defense in depth implements multiple layers of defense and countermeasures, making maximum use of certified equipment in each layer to facilitate system accreditation.

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Web-based Resource

A good online resource for system and software assurance is the US Department of Homeland Security's Build Security In [1] web site (DHS 2010), which provides resources for best practices, knowledge, and tools for engineering secure systems.

Discipline Standards

Information to be supplied at a later date.

Personnel Considerations

Information to be supplied at a later date.

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be supplied at a later date.

Practical Considerations

Pitfalls

Information to be provided at a later date.

Proven Practices

Information to be provided at a later date.

Other Considerations

Information to be provided at a later date.

References

Works Cited

Baldwin, K., J. Miller, P. Popick, and J. Goodnight. 2012. *The United States Department of Defense Revitalization of System Security Engineering Through Program Protection.* Proceedings of the 2012 IEEE Systems Conference, 19-22 March 2012, Vancouver, BC, Canada. Accessed 28 August 2012 at http://www.acq.osd.mil/se/docs/IEEE-SSE-Paper-02152012-Bkmarks.pdf [2].

CNSS. 2010. *National Information Assurance Glossary", Committee on National Security Systems Instruction (CNSSI) no. 4009". Fort Meade, MD, USA: The Committee on National Security Systems.*

DHS. 2010. *Build Security In.* Washington, DC, USA: US Department of Homeland Security (DHS). Accessed September 11, 2011. Available: https://buildsecurityin.us-cert.gov.

DODI5200.44, United States Department of Defense, *Protection of Mission Critical Functions to Achieve Trusted Systems and Networks*, Department of Defense Instruction Number 5200.44, November 2012, Accessed 3 November 2014 at Defense Technical Information Center http://www.dtic.mil/whs/directives/corres/pdf/520044p.pdf [3].

Goertzel, K., et al. 2008. *Enhancing the Development Life Cycle to Produce Secure Software: A Reference Guidebook on Software Assurance.* Washington, DC, USA: Data and Analysis Center for Software (DACS)/US Department of Homeland Security (DHS).

NATO. 2010. *Engineering for System Assurance in NATO programs.* Washington, DC, USA: NATO Standardization Agency. DoD 5220.22M-NISPOM-NATO-AEP-67.

Tipton, H.F. (ed.). 2006. *Official (ISC)2 guide to the CISSP CBK,* 1st ed. Boston, MA, USA: Auerbach Publications.

Primary References

Anderson, R.J. 2008. *Security Engineering: A Guide to Building Dependable Distributed Systems,* 2nd Ed. New York, NY, USA: John Wiley & Sons. Accessed October 24, 2014 at http://www.cl.cam.ac.uk/~rja14/book.html

DAU. 2012. "Defense Acquisition Guidebook (DAG): Chapter 13 -- Program Protection." Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). November 8, 2012. Accessed October 24, 2014 at https://dag.dau.mil/ [4]

ISO. 2008. "Information technology -- Security techniques -- Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®)," Second Edition. Geneva, Switzerland: International Organization for Standardization (ISO), ISO/IEC 21827:2008.

ISO/IEC. 2013. "Information technology — Security techniques — Information security management systems — Requirements," Second Edition. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 27001:2013.

Kissel, R., K. Stine, M. Scholl, H. Rossman, J. Fahlsing, J. Gulick. 2008. "Security Considerations in the System Development Life Cycle," Revision 2. Gaithersburg, MD. National Institute of Standard and Technology (NIST), NIST 800-64 Revision 2:2008. Accessed October 24, 2014 at the Computer Security Resource Center [5]

Ross, R., J.C. Oren, M. McEvilley. 2014. "Systems Security Engineering: An Integrated Approach to Building Trustworthy Resilient Systems." Gaithersburg, MD. National Institute of Standard and Technology (NIST) Special Publication (SP), NIST SP 800-160:2014 (Initial Public Draft). Accessed October 24, 2014 at the Computer Security Resource Center http://csrc.nist.gov/publications/drafts/800-160/sp800_160_draft.pdf [6]

Additional References

Allen, Julia; Barnum, Sean; Ellison, Robert; McGraw, Gary; and Mead, Nancy. 2008. *Software security engineering: a guide for project managers.* New York, NY, USA: Addison Wesley Professional.

ISO. 2005. *Information technology -- Security techniques -- Code of practice for information security management.* Geneva, Switzerland: International Organization for Standardization (ISO). ISO/IEC 27002:2005.

Jurjens, J. 2005. "Sound Methods and effective tools for model-based security engineering with UML." *Proceedings of the 2005 International Conference on Software Engineering.* Munich, GE: ICSE, 15-21 May.

MITRE. 2012. "Systems Engineering for Mission Assurance." In *Systems Engineering Guide.* Accessed 19 June 2012 at MITRE http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ se_for_mission_assurance/ [7].

NIST SP 800-160. *Systems Security Engineering - An Integrated Approach to Building Trustworthy Resilient Systems*. National Institute of Standards and Technology, U.S. Department of Commerce, Special Publication 800-160. Accessed October 24, 2014 at the Computer Security Resource Center http://csrc.nist.gov/publications/ drafts/800-160/sp800_160_draft.pdf [6].

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] https://buildsecurityin.us-cert.gov/
[2] http://www.acq.osd.mil/se/docs/IEEE-SSE-Paper-02152012-Bkmarks.pdf
[3] http://www.dtic.mil/whs/directives/corres/pdf/520044p.pdf
[4] https://dag.dau.mil/
[5] http://csrc.nist.gov/publications/nistpubs/800-64-Rev2/SP800-64-Revision2.pdf
[6] http://csrc.nist.gov/publications/drafts/800-160/sp800_160_draft.pdf
[7] http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/se_for_mission_assurance/

Electromagnetic Interference/Electromagnetic Compatibility

Electromagnetic Interference (EMI) is the disruption of operation of an electronic device when it is in the vicinity of an electromagnetic field in the radio frequency (RF) spectrum. Many electronic devices fail to work properly in the presence of strong RF fields. The disturbance may interrupt, obstruct, or otherwise degrade or limit the effective performance of the circuit. The source may be any object, artificial or natural, that carries rapidly changing electrical currents.

Electromagnetic Compatibility (EMC) is the ability of systems, equipment, and devices that utilize the electromagnetic spectrum to operate in their intended operational environments without suffering unacceptable degradation or causing unintentional degradation because of electromagnetic radiation or response. It involves the application of sound electromagnetic spectrum management; system, equipment, and device design configuration that ensures interference-free operation; and clear concepts and doctrines that maximize operational effectiveness (DAU 2010, Chapter 7).

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Spectrum

Each nation has the right of sovereignty over the use of its spectrum and must recognize that other nations reserve the same right. It is essential that regional and global forums exist for the discussion and resolution of spectrum development and infringement issues between bordering and proximal countries that might otherwise be difficult to resolve.

The oldest, largest, and unquestionably the most important such forum, with 193 member countries, is the International Telecommunications Union (ITU) agency of the United Nations, which manages spectrum at a global level. As stated in Chapter 3 of the NTIA Manual, “The International Telecommunication Union (ITU)...is responsible for international frequency allocations, worldwide telecommunications standards and telecommunication development activities” (NTIA 2011, 3-2). The broad functions of the ITU are the regulation, coordination and development of international telecommunications.

The spectrum allocation process is conducted by many different international telecommunication geographical committees. Figure 1 shows the various international forums represented worldwide.



Figure 1. International & Regional Spectrum Management Forums. (SEBoK Original)

Assigning frequencies is very complicated, as shown in the radio spectrum allocation chart in Figure 2. Sometimes, commercial entities try to use frequencies that are actually assigned to US government agencies, such as the Department of Defense (DoD). One such incident occurred when an automatic garage door vendor installed doors on homes situated near a government installation. Random opening and closing of the doors created a problem for the vendor that could have been avoided.

Four ITU organizations affect spectrum management (Stine and Portigal 2004):

1. World Radio-communication Conference (WRC)
2. Radio Regulations Board (RRB)
3. Radio-communications Bureau (RB)
4. Radio-communication Study Groups (RSG)

The WRC meets every four years to review and modify current frequency allocations. The RB registers frequency assignments and maintains the master international register. The RRB approves the Rules of Procedures used by the BR to register frequency assignments and adjudicates interference conflicts among member nations. The SG analyzes spectrum usage in terrestrial and space applications and makes allocation recommendations to the WRC. Most member nations generally develop national frequency allocation polices that are consistent with the Radio Regulations (RR). These regulations have treaty status.

Dual Management of Spectrum in the US

Whereas most countries have a single government agency to perform the spectrum management function, the US has a dual management scheme intended to insure that

- decisions concerning commercial interests are made only after considering their impact on government systems; and
- government usage supports commercial interests.

The details of this scheme, established by the Communications Act of 1934, are as follows:

- the Federal Communications Commission (FCC) is responsible for all non-government usage
- the FCC is directly responsible to Congress;
- the president is responsible for federal government usage, and by executive order, delegates the federal government spectrum management to the National *Telecommunications and Information Administration (NTIA); and
- the NTIA is under the authority of the Secretary of Commerce.

The FCC regulates all non-federal government telecommunications under Title 47 of the Code of Federal Regulations. For example, see FCC (2009, 11299-11318). The FCC is directed by five Commissioners appointed by the president and confirmed by the Senate for five-year terms. The Commission staff is organized by function. The responsibilities of the six operating Bureaus include processing applications for licenses, analyzing complaints, conducting investigations, implementing regulatory programs, and conducting hearings (http://www.fcc.gov).

The NTIA performs spectrum management function through the Office of Spectrum Management (OSM), governed by the Manual of Regulations and Procedures for Federal Radio Frequency Management. The IRAC develops and executes policies, procedures, and technical criteria pertinent to the allocation, management, and usage of spectrum. The Spectrum Planning and Policy Advisory Committee (SPAC) reviews the reviews IRAC plans, balancing considerations of manufacturing, commerce, research, and academic interests.

Within the DoD, spectrum planning and routine operation activities are cooperatively managed. Spectrum certification is a mandated process designed to ensure that

1. frequency band usage and type of service in a given band are in conformance with the appropriate national and international tables of frequency allocations;
2. equipment conforms to all applicable standards, specifications, and regulations; and
3. approval is provided for expenditures to develop equipment dependent upon wireless communications.

Host Nation Coordination and Host Nation Approval

In peacetime, international spectrum governance requires military forces to obtain host nation permission — Host Nation Coordination (HNC)/Host Nation Approval (HNA) — to operate spectrum-dependent systems and equipment within a sovereign nation. For example, international governance is honored and enforced within the United States by the US departments of State, Defense, and the user service.

In wartime, international spectrum governance is not honored between warring countries; however, the sovereign spectrum rights of bordering countries must be respected by military forces executing their assigned missions. For example, HNA is solicited by US naval forces to use spectrum-dependent systems and equipment in bordering countries' airspace and/or on bordering countries' soil. HNA must be obtained before the operation of spectrum-dependent systems and equipment within a sovereign nation. The combatant commander is responsible for coordinating requests with sovereign nations within his or her area of responsibility. Because the combatant commander has no authority over a sovereign nation, the HNC/HNA process can be lengthy and needs to be started early in the development of a system. Figure 2 illustrates a spectrum example.

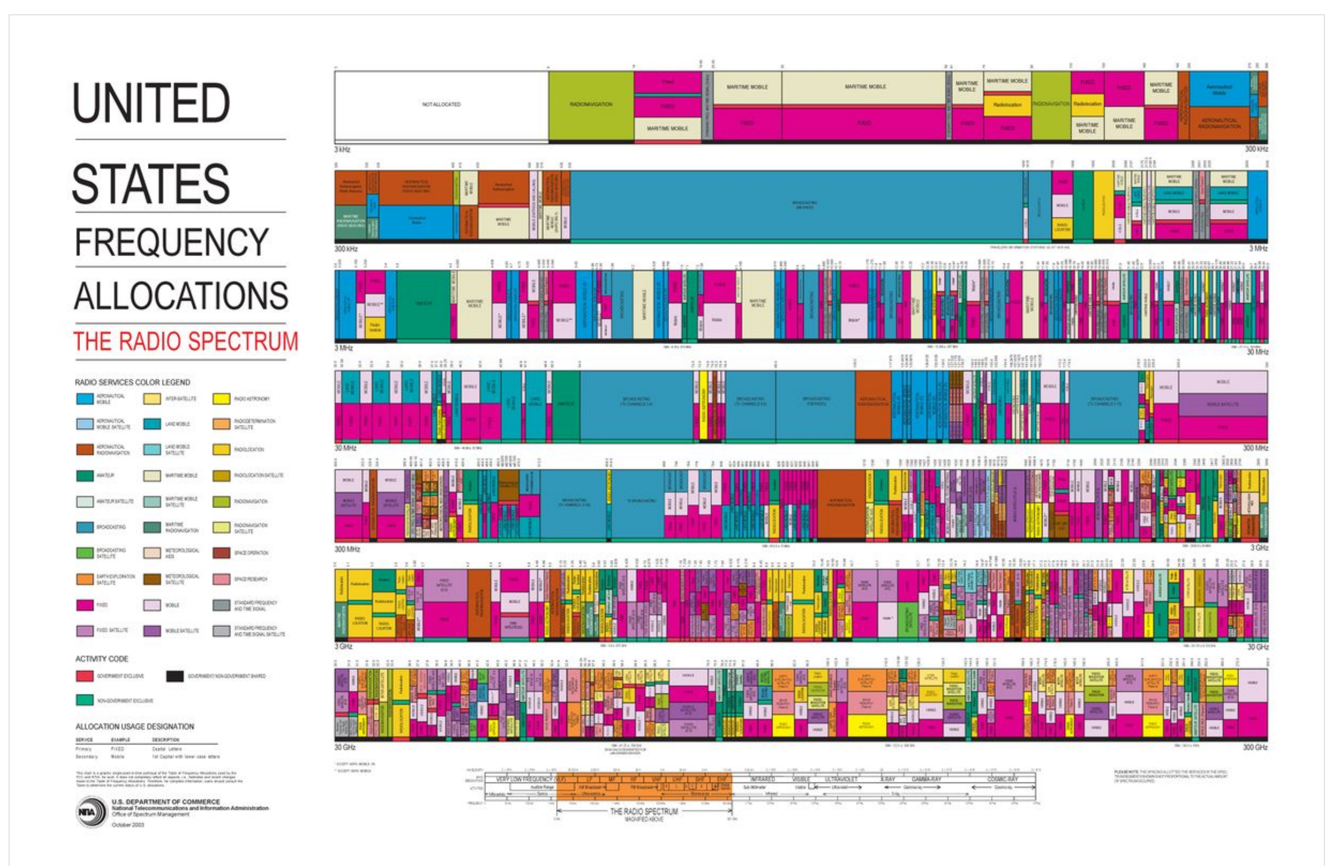


Figure 2. The Radio Spectrum (Department of Commerce 2003). Released by the U.S. Department of Commerce. Source is available at http://www.ntia.doc.gov/files/ntia/publications/2003-allochrt.pdf (Retrieved September 15, 2011)

Practical Considerations

EMI/EMC is difficult to achieve for systems that operate world-wide because of the different frequencies in which products are designed to operate in each of the telecommunication areas. Billions of US dollars have been spent in retrofitting US DoD equipment to operate successfully in other countries.

It is important to note that the nuclear radiation environment is drastically more stressing than, and very different from, the space radiation environment.

System Description

Narrowband and Broadband Emissions

To help in analyzing conducted and radiated interference effects, EMI is categorized into two types—narrowband and broadband—which are defined as follows:

- **Narrowband Emissions**

  A narrowband signal occupies a very small portion of the radio spectrum… Such signals are usually continuous sine waves (CW) and may be continuous or intermittent in occurrence… Spurious emissions, such as harmonic outputs of narrowband communication transmitters, power-line hum, local oscillators, signal generators, test equipment, and many other man made sources are narrowband emitters. (Bagad 2009, G-1)

- **Broadband Emissions**

  A broadband signal may spread its energy across hundreds of megahertz or more… This type of signal is composed of narrow pulses having relatively short rise and fall times. Broadband signals are further

> *divided into random and impulse sources. These may be transient, continuous or intermittent in occurrence. Examples include unintentional emissions from communication and radar transmitters, electric switch contacts, computers, thermostats, ignition systems, voltage regulators, pulse generators, and intermittent ground connections.* (Bagad 2009, G-1)

TEMPEST

TEMPEST is a codename used to refer to the field of emission security. The National Security Agency (NSA) investigations conducted to study compromising emission (CE) were codenamed TEMPEST. National Security Telecommunications Information Systems Security Issuance (NSTISSI)-7000 states:

> *Electronic and electromechanical information-processing equipment can produce unintentional intelligence-bearing emanations, commonly known as TEMPEST. If intercepted and analyzed, these emanations may disclose information transmitted, received, handled, or otherwise processed by the equipment.* (NSTISS 1993, 3)

These compromising emanations consist of electrical, mechanical, or acoustical energy intentionally or unintentionally emitted by sources within equipment or systems which process national security information. Electronic communications equipment needs to be secured from potential eavesdroppers while allowing security agencies to intercept and interpret similar signals from other sources. The ranges at which these signals can be intercepted depends upon the functional design of the information processing equipment, its installation, and prevailing environmental conditions.

Electronic devices and systems can be designed, by means of Radiation Hardening techniques, to resist damage or malfunction caused by ionizing and other forms of radiation (Van Lint and Holmes Siedle 2000). Electronics in systems can be exposed to ionizing radiation in the Van Allen radiation belts around the Earth's atmosphere, cosmic radiation in outer space, gamma or neutron radiation near nuclear reactors, and electromagnetic pulses (EMP) during nuclear events.

A single charged particle can affect thousands of electrons, causing electronic noise that subsequently produces inaccurate signals. These errors could affect safe and effective operation of satellites, spacecraft, and nuclear devices. Lattice displacement is permanent damage to the arrangement of atoms in element crystals within electronic devices. Lattice displacement is caused by neutrons, protons, alpha particles, and heavy ions. Ionization effects are temporary damages that create latch-up glitches in high power transistors and soft errors like bit flips in digital devices. Ionization effects are caused by charged particles.

Most radiation-hardened components are based on the functionality of their commercial equivalents. Design features and manufacturing variations are incorporated to reduce the components' susceptibility to interference from radiation. Physical design techniques include insulating substrates, package shielding, chip shielding with depleted boron, and magneto-resistive RAM. Logical design techniques include error-correcting memory, error detection in processing paths, and redundant elements at both circuit and subsystem levels (Dawes 1991). Nuclear hardness is expressed as susceptibility or vulnerability for given environmental conditions. These environmental conditions include peak radiation levels, overpressure, dose rates, and total dosage.

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Information to be supplied at a later date.

Discipline Standards

Information to be supplied at a later date.

Personnel Considerations

Information to be supplied at a later date.

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be provided at a later date.

Practical Considerations

Pitfalls

Information to be provided at a later date.

Proven Practices

Information to be provided at a later date.

Other Considerations

Information to be provided at a later date.

References

Works Cited

Bagad, V.S. 2009. *Electronic Product Design,* 4th ed. Pune, India: Technical Publications Pune.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

NSTISS. 1993. *Tempest Countermeasures for Facilities.* Ft. Meade, MD, USA: National Security Telecommunications and Information Systems Security (NSTISSI). 29 November, 1993. NSTISSI No. 7000.

NTIA. 2011. *Manual of Regulations and Procedures for Federal Radio Frequency Management,* May 2011 Revision of the 2008 Edition. Washington, DC, USA: National Telecommunications and Information Administration, U.S. Department of Commerce.

Stine, J. and D. Portigal. 2004. *An Introduction to Spectrum Management.* Bedford, MA, USA: MITRE Technical Report Spectrum. March 2004.

Van Lint, V.A.J. and A.G. Holmes Siedle. 2000. *Radiation Effects in Electronics: Encyclopedia of Physical Science and Technology.* New York, NY, USA: Academic Press.

Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

System Resilience

According to the Oxford English Dictionary on Historical Principles (1973), resilience is “the act of rebounding or springing back.” This definition most directly fits the situation of materials which return to their original shape after deformation. For human-made, or engineered systems the definition of resilience (glossary) can be extended to include the ability to maintain capability (glossary) in the face of a disruption (glossary). The US government definition for resilient infrastructure (glossary) systems is the "ability of systems, infrastructures, government, business, communities, and individuals to resist, tolerate, absorb, recover from, prepare for, or adapt to an adverse occurrence that causes harm, destruction, or loss of national significance” (DHS 2010).

The name **Resilience Engineering** was coined in the book Resilience Engineering: Concepts and Precepts (Hollnagel et al 2006). The authors make clear in this book that Resilience Engineering has to do with the resilience of the organisations that design and operate engineered systems and not with the systems themselves The term **System Resilience** used in this article is primarily concerned with the techniques used to consider the resilience of engineered systems directly. To fully achieve this SE also needs to consider the resilience of those organisational and human systems which enable the life cycle of an engineered system. The techniques or design principles used to assess and improve the resilience of engineered systems across their Life Cycle (glossary) are elaborated by Jackson and Ferris (2013).

Overview

Resilience is a relatively new term in the SE realm, appearing only in the 2006 time frame and becoming popularized in 2010. The recent application of “resilience” to engineered systems has led to confusion over its meaning and a proliferation of alternative definitions. (One expert claims that well over 100 unique definitions of resilience have appeared.) While the details of definitions will continue to be discussed and debated, the information here should provide a working understanding of the meaning and implementation of resilience, sufficient for a system engineer to effectively address it.

Definition

It is difficult to identify a single definition that – word for word – satisfies all. However, it is possible to gain general agreement of what is meant by resilience of engineered systems; viz., resilience is the ability to provide required capability (glossary) in the face of adversity.

Scope of the Means

In applying this definition, one needs to consider the range of means by which resilience is achieved: The means of achieving resilience include avoiding, withstanding, recovering from and evolving and adapting to adversity. These may also be considered the fundamental objectives of resilience, Brtis (2013). Classically, resilience includes “withstanding” and “recovering” from adversity. For the purpose of engineered systems, “avoiding” adversity is considered a legitimate means of achieving resilience. Jackson and Ferris (2016). Also, it is believed that resilience should consider the system’s ability to “evolve and adapt” to future threats and unknown-unknowns.

Scope of the Adversity

Adversity is any condition that may degrade the desired capability of a system. We propose that the SE must consider all sources and types of adversity; e.g., from environmental sources, due to normal failure, as well as from opponents, friendlies and neutral parties. Adversity from human sources may be malicious or accidental. Adversities may be expected or not. Adversity may include "unknown unknowns." The techniques for achieving resilience discussed below are applicable to both hostile and non-hostile adversities in both civil and military domains.

Non-hostile adversities will dominate in the civil domain and hostile adversities will predominate in the military domain.

Notably, a single incident may be the result of multiple adversities, such as a human error committed in the attempt to recover from another adversity.

Jackson & Ferris Taxonomy

Figure 1 depicts the loss and recovery of the functionality of a system. In the taxonomy proposed by Jackson and Ferris (2013) four attributes can lead to a resilient system and may posess four attributes: robustness (glossary), adaptability (glossary), tolerance (glossary), and integrity (glossary) — and fourteen design techniques and 20 support techniques that can achieve these attributes. These four attributes are adapted from Hollnagel, Woods, and Leveson (2006), and the design techniques are extracted from Hollnagel et al. and are elaborated based on Jackson and Ferris (2013) for civil systems.

Other sources for example, DHS (2010) lists the following additional attributes: rapidly, affordability and learning capacity.

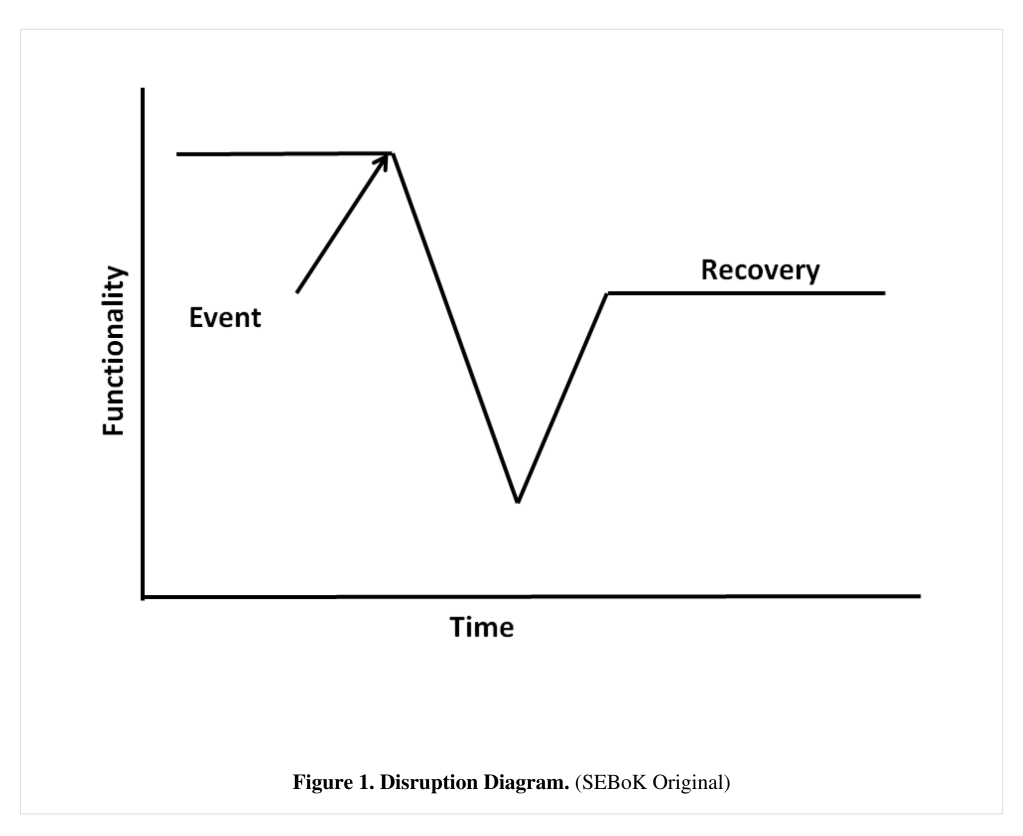


Figure 1. Disruption Diagram. (SEBoK Original)

The Robustness Attribute

Robustness is the attribute of a system that allows it to withstand a threat in the normal operating state. Resilience allows that the capacity of a system may be exceeded, forcing the system to rely on the remaining attributes to achieve recovery. The following design techniques tend to achieve robustness:

- absorption (glossary)
- physical redundancy (glossary)
- functional redundancy (glossary)

The Adaptability Attribute

Adaptability is the attribute of a system that allows it to restructure itself in the face of a threat. Adaptability can apply to any phase of the event including detecting and avoiding the adversity and restructuring to return to normal operation. The following design techniques apply to the adaptability attribute:

- restructuring (glossary)
- human in the loop (glossary)
- complexity avoidance (glossary)
- drift correction (glossary)

The Tolerance Attribute

Tolerance is the attribute of a system that allows it to degrade gracefully following an encounter with adversity. The following design techniques apply to the tolerance attribute.

- modularity (glossary)
- loose Coupling (glossary)
- neutral state (glossary)
- reparability (glossary)
- defense in depth (glossary)

The Integrity Attribute

Integrity is the attribute of a system that allows it to operate before, during, and after an encounter with a threat. Integrity is also called cohesion which according to (Hitchins 2009), a basic characteristic of a system. The following global design technique applies to the integrity attribute.

- Internode Interaction (glossary)

MITRE Taxonomy

Brits (2016) builds on the cyber resilience work of Bodeau & Graubart (2011) and proposes an objectives-based three layer taxonomy for thinking about resilience. The three layers include: 1) four fundamental objectives of resilience, 2) ten means objectives of resilience, and, 3) 23 engineering techniques for achieving resilience.

The four fundamental objectives, which identify the intrinsic value of resilience, are:

- Avoid adversity
- Withstand adversity
- Recover from adversity
- Evolve and adapt to unanticipated adversity

The ten means objectives are not ends in themselves - as are the fundamental objective - but do tend to result in the achievement of the fundamental objectives: The means objectives are:

- anticipate
- understand
- disaggregate
- prepare
- prevent
- continue
- constrain
- redeploy
- transform
- re-architect

The 23 engineering techniques that tend to achieve the fundamental objectives are:

- adaptive response
- analytic monitoring
- coordinated defense
- deception
- distribution

- detection avoidance
- diversification
- dynamic positioning
- dynamic representation
- effect tolerance
- non-persistence
- privilege restriction
- proliferation
- protection
- realignment
- reconfiguring
- redundancy
- replacement
- segmentation
- substantiated integrity
- substitution
- threat suppression
- unpredictability

The relationships between the three layers of this taxonomy are many-to-many relationships.

The Jackson & Ferris taxonomy comes from the civil resilience perspective and the Brtis/MITRE taxonomy comes from the military perspective. Jackson and Brtis (2017) have shown that many of the techniques of the two taxonomies are equivalent and that some techniques are unique to each domain.

Techniques for Achieving Resilience

Techniques for achieving resilience have been identified for both the civil and military domains. Jackson and Ferris (2013) have identified techniques for the civil domain and Brtis (2016) has identified techniques for the military domain. There is overlap between these two sets and also some differences. Jackson and Brtis (2017) compare the techniques in the two domains and show their commonalities and their differences.

Techniques for the Civil Domain

34 techniques and support techniques for the civil domain described by Jackson and Ferris (2013) include both design and process techniques that will be used to define a system of interest in an effort to make it resilient. These techniques were extracted from many sources. Prominent among these sources is Hollnagel et al (2006). Other sources include Leveson (1995), Reason (1997), Perrow (1999), and Billings (1997). Some techniques were implied in case study reports, such as the 9/11 Commission report (2004) and the US-Canada Task Force report (2004) following the 2003 blackout. These techniques include very simple and well-known techniques as physical redundancy and more sophisticated techniques as loose coupling. Some of these techniques are domain dependent, such as loose coupling, which is important in the power distribution domain as discussed by Perrow (1999). These techniques will be the input to the state model of Jackson, Cook, and Ferris (2015) to determine the characteristics of a given system for a given threat. In the resilience literature the term technique is used to describe both scientifically accepted techniques and also heuristics, design rules determined from experience as described by Rechtin (1991). Jackson and Ferris (2013) showed that it is often necessary to invoke these techniques in combinations to best enhance resilience. This concept is called defense in depth. Pariès (2011) illustrates how defense in depth was used to achieve resilience in the 2009 ditching of US Airways Flight 1549. Uday and Marais (2015) apply the above techniques to the design of a system-of-systems. Henry and Ramirez-Marquez (2016) describe the state of the U.S. East Coast infrastructure in resilience terms following the impact of Hurricane Sandy in 2012. Bodeau & Graubert (2011) propose a framework for understanding and addressing cyber-resilience. They propose a taxonomy comprised

of four goals, eight objectives, and fourteen cyber-resilience techniques. Many of these goals, objectives and practices can be applied to non-cyber resilience. Jackson and Ferris (2013) have collected 14 design techniques from various authoritative sources. These techniques are applicable primarily to civil systems including civil infrastructure, aviation, and power grids. In addition to the 14 design techniques, Jackson and Ferris (2013) also identify 20 support techniques that are narrower in scope than the above design techniques.

Techniques for the Military Domain

Brtis (2016), in the third level of his taxonomy discussed above identifies 23 engineering techniques for achieving resilience in the military domain. Jackson and Brtis (2017) have shown that many of the civil and military techniques are equivalent though some are unique to each domain.

The Resilience Process

Implementation of resilience in a system requires the execution of both analytic and holistic processes. In particular, the use of architecting with the associated heuristics is required. Inputs are the desired level of resilience and the characteristics of a threat or disruption. Outputs are the characteristics of the system, particularly the architectural characteristics and the nature of the elements (e.g., hardware, software, or humans). Artifacts depend on the domain of the system. For technological systems, specification and architectural descriptions will result. For enterprise systems, enterprise plans will result. Both analytic and holistic methods, including the techniques of architecting, are required. Analytic methods determine required robustness. Holistic methods determine required adaptability, tolerance, and integrity. One pitfall is to depend on just a single technique to achieving resilience. The technique of defense in depth suggests that multiple techniques may be required to achieve resilience.

Practical Considerations

Resilience is difficult to achieve for infrastructure systems because the nodes (cities, counties, states, and private entities) are reluctant to cooperate with each other. Another barrier to resilience is cost. For example, achieving redundancy in dams and levees can be prohibitively expensive. Other aspects, such as communicating on common frequencies, can be low or moderate cost; even there, cultural barriers have to be overcome for implementation.

System Description

System resilience is the ability of a Engineered System (glossary) to provide required capability (glossary) in the face of adversity (glossary). Resilience in the realm of systems engineering involves identifying: 1) the capabilities that are required of the system, 2) the adverse conditions under which the system is required to deliver those capabilities, and 3) the systems engineering to ensure that the system can provide the required capabilities.

Put simply, resilience is achieved by a systems engineering focus on adverse conditions.

Resilience of Processes

It is important to recognize that processes are systems – in fact Systems Engineering is a system. Discussions relating to the resilience of such “process” systems include seven key resiliencies that successful sociotechnical systems intending to accomplish system engineering must have, Warfield (2008). Ashby’s Law of Requisite Variety and Pareto’s Law of Requisite Saliency are the most familiar. The scope and time of arrival of Contract Change Orders that require system engineering attention pose significant risk. Ones that occur during detailed design that affect the requirements baseline and system design baseline and occur faster than can be accommodated are particularly threatening.

Discipline Management

Most enterprises, both military and commercial, include organizations generally known as Advanced Design. These organizations are responsible for defining the architecture of a system at the very highest level of the system architecture. This architecture reflects the resilience techniques described in Jackson and Ferris (2013) and Brtis (2016) and the processes associated with that system. In many domains, such as fire protection, no such organization will exist. However, the system architecture will still need to be defined by the highest level of management in that organization. In addition, some aspects of resilience will be established by government imposed requirements.

Discipline Relationships

Interactions

Outputs

The primary outputs of the resilience discipline are a subset of the principles described by Jackson and Ferris (2013) which have been determined to be appropriate for a given system, threat, and desired state of resilience as determined by the state-transition analysis described below. The processes requiring these outputs are the system design and system architecture processes.

Inputs

Inputs to the state model described in Jackson, Cook, and Ferris (2015) include (1) type of system of interest, (2) nature of threats to the system (earthquakes, terrorist threats, human error, etc.) (3) techniques for potential architectural design, and (4) predicted probability of success of individual techniques.

Dependencies

The techniques identified for the achieving resilience may also be used by other systems engineering areas of concern such as safety, reliability, human factors, availability, maintainability, human factors, security, and others. For example, the physical redundancy technique may help achieve resilience, reliability, and safety. Resilience design and analysis should be conducted in concert with the various ‘ilities. The goal being to create a system which -- from the beginning -- meets the requirements for resilience and other ‘ilities.

Discipline Standards

ASIS (2009) has published a standard pertaining to the resilience of organizational systems.

NIST 800-160 considers resilience of physical systems.

Personnel Considerations

Humans are important components of systems for which resilience is desired. This aspect is reflected in the human in the loop technique identified by Jackson and Ferris (2013). Decisions made by the humans are at the discretion of the humans in real time. Apollo 11 described by Eyles (2009) is a good example.

Metrics

Uday & Marais (2015) performed a survey of resilience metrics. Those identified include:

- Time duration of failure
- Time duration of recovery
- Ratio of performance recovery to performance loss
- A function of speed of recovery
- Performance before and after the disruption and recovery actions
- System importance measures

Jackson (2016) developed a metric to evaluate various systems in four domains: aviation, fire protection, rail, and power distribution, for the principles that were lacking in ten different case studies. The principles are from the set identified by Jackson and Ferris (2013) and are represented in the form of a histogram plotting principles against frequency of omission. The data in these gaps were taken from case studies in which the lack of principles was inferred from recommendations by domain experts in the various cases cited.

Brtis (2016) surveyed and evaluated a number of potential resilience metrics and identified the following: [Note: This reference is going through approval for public release and should be referenceable by the end of July 2016.]

- Maximum outage period
- Maximum brownout period
- Maximum outage depth
- Expected value of capability: the probability-weighted average of capability delivered
- Threat resiliency (the time integrated ratio of the capability provided divided by the minimum needed capability)
- Expected availability of required capability (the likelihood that for a given adverse environment the required capability level will be available)
- Resilience levels (the ability to provide required capability in a hierarchy of increasingly difficult adversity)
- Cost to the opponent
- Cost-benefit to the opponent
- Resource resiliency (the degradation of capability that occurs as successive contributing assets are lost)

Brtis found that multiple metrics may be required, depending on the situation. However, if one had to select a single most effective metric for reflecting the meaning of resilience, it would be the expected availability of the required capability. Expected availability of the required capability is the probability-weighted sum of the availability summed across the scenarios under consideration. In its most basic form, this metric can be represented mathematically as:

where,

R = Resilience of the required capability (Cr);

n = the number of exhaustive and mutually exclusive adversity scenarios within a context (n can equal 1);

Pi = the probability of adversity scenario I;

Cr(t)_i = timewise availability of the required capability during scenario I; --- 0 if below the required level --- 1 if at or above the required value (Where circumstances dictate this may take on a more complex, non-binary function of time.);

T = length of the time of interest.

Models

The state-transition model described by Jackson et al (2015) describes a system in its various states before, during, and after an encounter with a threat. The model identifies seven different states as the system passes from a nominal operational state to minimally acceptable functional state as shown in the figure below. In addition, the model identifies 28 transition paths from state to state. To accomplish each transition the designer must invoke one or more of the 34 principles or support principles described by Jackson and Ferris (2013). The designs implied by these principles can then be entered into a simulation to determine the total effectiveness of each design.

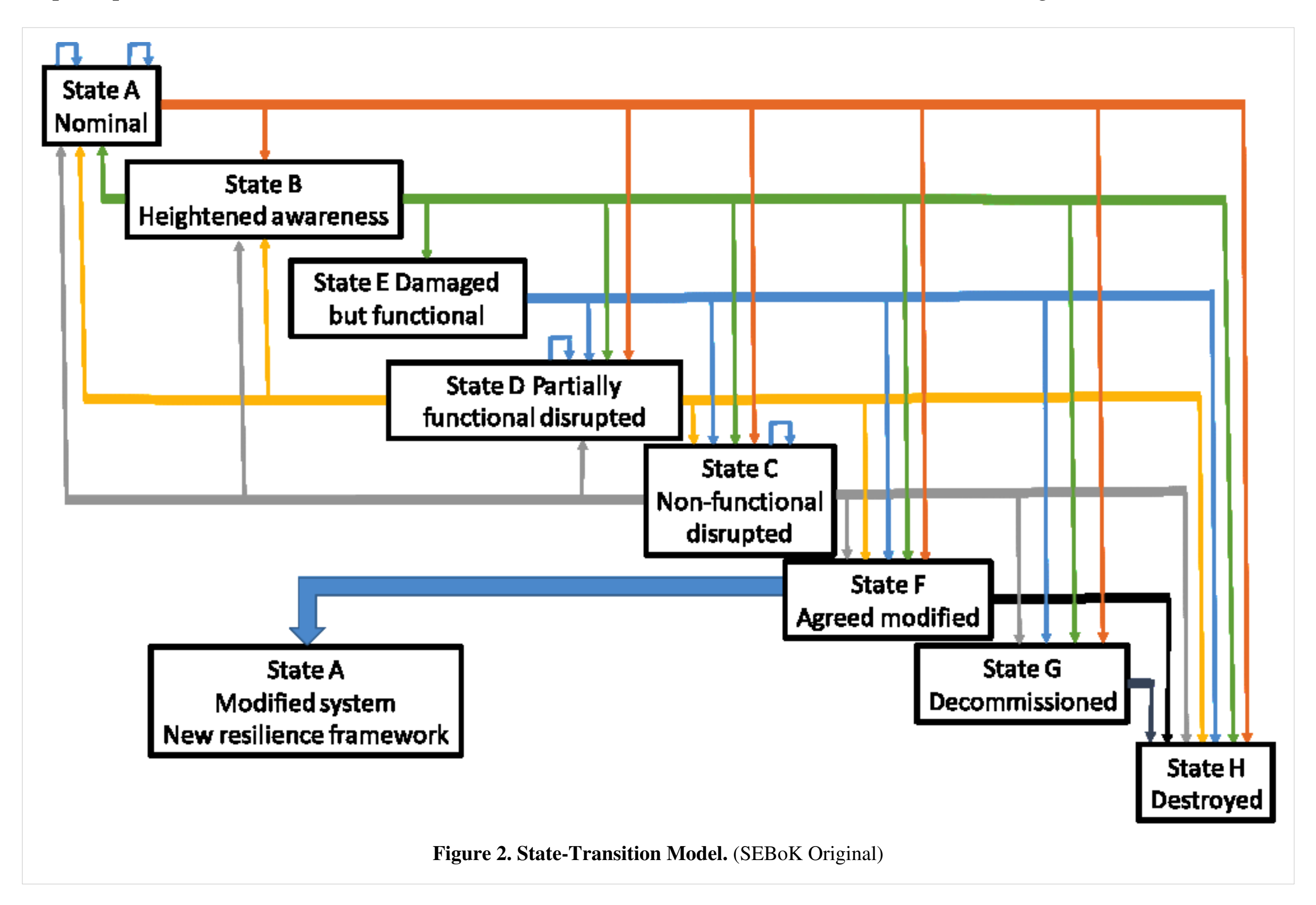


Figure 2. State-Transition Model. (SEBoK Original)

Tools

No tools dedicated to resilience have been identified.

Practical Considerations

Pitfalls

Information to be provided at a later date.

Proven Practices

Information to be provided at a later date.

Other Considerations

Information to be provided at a later date.

References

Works Cited

9/11 Commission. (2004). 9/11 Commission Report.

ASIS International. (2009). Organizational Resilience: Security, Preparedness, and Continuity Management Systems--Requirements With Guidance for Use. Alexandria, VA, USA: ASIS International.

Billings, C. (1997). Aviation Automation: The Search for Human-Centered Approach. Mahwah, NJ: Lawrence Erlbaum Associates.

Bodeau, D. K, & Graubart, R. (2011). Cyber Resiliency Engineering Framework, MITRE Technical Report #110237, The MITRE Corporation.

Brtis, J. S. (2016). How to Think About Resilience, MITRE Technical Report, MITRE Corporation.

Hollnagel, E., D. Woods, and N. Leveson (eds). 2006. *Resilience Engineering: Concepts and Precepts.* Aldershot, UK: Ashgate Publishing Limited.

INCOSE (2015). Systems Engineering Handbook, a Guide for System Life Cycle Processes and Activities. San Diego, Wiley.

Jackson, S., & Ferris, T. (2013). Resilience Principles for Engineered Systems. Systems Engineering, 16(2), 152-164.

Jackson, S., Cook, S. C., & Ferris, T. (2015). A Generic State-Machine Model of System Resilience. Insight, 18.

Jackson, S. & Ferris, T. L. (2016). Proactive and Reactive Resilience: A Comparison of Perspectives.

Jackson, W. S. (2016). Evaluation of Resilience Principles for Engineered Systems. Unpublished PhD, University of South Australia, Adelaide, Australia.

Leveson, N. (1995). Safeware: System Safety and Computers. Reading, Massachusetts: Addison Wesley.

Madni, Azad,, & Jackson, S. (2009). Towards a conceptual framework for resilience engineering. Institute of Electrical and Electronics Engineers (IEEE) Systems Journal, 3(2), 181-191.

Pariès, J. (2011). Lessons from the Hudson. In E. Hollnagel, J. Pariès, D. D. Woods & J. Wreathhall (Eds.), Resilience Engineering in Practice: A Guidebook. Farnham, Surrey: Ashgate Publishing Limited.

Perrow, C. (1999). Normal Accidents: Living With High Risk Technologies. Princeton, NJ: Princeton University Press.

Reason, J. (1997). Managing the Risks of Organisational Accidents. Aldershot, UK: Ashgate Publishing Limited.

Rechtin, E. (1991). Systems Architecting: Creating and Building Complex Systems. Englewood Cliffs, NJ: CRC Press.

US-Canada Power System Outage Task Force. (2004). Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations. Washington-Ottawa.

Uday, P., & Morais, K. (2015). Designing Resilient Systems-of-Systems: A Survey of Metrics, Methods, and Challenges. Systems Engineering, 18(5), 491-510.

Warfield, J. N. (2008) "A Challenge for Systems Engineers: To Evolve Toward Systems Science" INCOSE INSIGHT, Volume 11, Issue 1, January 2008.

Primary References

Hollnagel, E., Woods, D. D., & Leveson, N. (Eds.). (2006). Resilience Engineering: Concepts and Precepts. Aldershot, UK: Ashgate Publishing Limited.

Jackson, S., & Ferris, T. (2013). Resilience Principles for Engineered Systems. Systems Engineering, 16(2), 152-164.

Jackson, S., Cook, S. C., & Ferris, T. (2015). Towards a Method to Describe Resilience to Assist in System Specification. Paper presented at the INCOSE Systems 2015.

Jackson, S.: Principles for Resilient Design - A Guide for Understanding and Implementation. Available at https://www.irgc.org/irgc-resource-guide-on-resilience Accessed 18th August 2016

Madni, Azad,, & Jackson, S. (2009). Towards a conceptual framework for resilience engineering. Institute of Electrical and Electronics Engineers (IEEE) Systems Journal, 3(2), 181-191.

Additional References

ASIS International. 2009. Organisational Resilience: Security, Preparedness, and Continuity Management Systems--Requirements With Guidance for Use. Alexandria, VA, USA: ASIS International.

Billings, Charles. 1997. Aviation Automation: The Search for Human-Centered Approach. Mahwah, NJ: Lawrence Erlbaum Associates.

Bodeu, D. K., and R. Graubart. 2011. Cyber Resiliency Engineering Framework.

Eyles, Don. 2009. "1202 Computer Error Almost Aborted Lunar Landing." Massachusetts Intitute of Technology, MIT News, accessed 6 April http://njnnetwork.com/2009/07/1202-computer-error-almost-aborted-lunar-landing/

Henry, Devanandham, and Emmanual Ramirez-Marquez. 2016. "On the Impacts of Power Outages during Hurrican Sandy -- A Resilience Based Analysis." Systems Engineering 19 (1):59-75.

OED. 1973. The Shorter Oxford English Dictionary on Historical Principles. edited by C. T. Onions. Oxford: Oxford Univeristy Press. Original edition, 1933.

Pariès, Jean. 2011. "Lessons from the Hudson." In Resilience Engineering in Practice: A Guidebook, edited by Erik Hollnagel, Jean Pariès, David D. Woods and John Wreathhall, 9-27. Farnham, Surrey: Ashgate Publishing Limited.

Rechtin, Eberhardt. 1991. Systems Architecting: Creating and Building Complex Systems. Englewood Cliffs, NJ: CRC Press.

Uday, Payuna, and Karen Morais. 2015. "Designing Resilient Systems-of-Systems: A Survey of Metrics, Methods, and Challenges." Systems Engineering 18 (5):491-510.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Manufacturability and Producibility

Manufacturability and producibility is an engineering specialty. The machines and processes used to build a system must be architected and designed. A systems engineering approach to manufacturing and production is necessary because manufacturing equipment and processes can sometimes cost more than the system being built (Maier and Rechtin 2002). Manufacturability and producibility can be a discriminator between competing system solution concepts and therefore must be considered early in the study period, as well as during the maturing of the final design solution.

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

The system being built might be intended to be one-of-a-kind, or to be reproduced multiple times. The manufacturing system differs for each of these situations and is tied to the type of system being built. For example, the manufacture of a single-board computer would be vastly different from the manufacture of an automobile. Production involves the repeated building of the designed system. Multiple production cycles require the consideration of production machine maintenance and downtime.

Manufacturing and production engineering involve similar systems engineering processes specifically tailored to the building of the system. Manufacturability and producibility are the key attributes of a system that determine the ease of manufacturing and production. While manufacturability is simply the ease of manufacture, producibility also encompasses other dimensions of the production task, including packaging and shipping. Both these attributes can be improved by incorporating proper design decisions that take into account the entire system life cycle (Blanchard and Fabrycky 2005).

System Description

Information to be supplied at a later date.

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Information to be supplied at a later date.

Discipline Standards

Information to be supplied at a later date.

Personnel Considerations

Information to be supplied at a later date.

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be supplied at a later date.

Practical Considerations

Pitfalls

Information to be provided at a later date.

Proven Practices

Information to be provided at a later date.

Other Considerations

Information to be provided at a later date.

References

Works Cited

Maier, M., and E. Rechtin. 2002. *The Art of Systems Architecting,* 2nd ed. Boca Raton, FL, USA: CRC Press.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englwood Cliffs, NJ, USA: Prentice-Hall.

Primary References

None.

Additional References

Anderson, D. 2010. *Design for Manufacturability & Concurrent Engineering; How to Design for Low Cost, Design in High Quality, Design for Lean Manufacture, and Design Quickly for Fast Production.* Cambria, CA, USA: CIM Press.

Boothroyd, G., P. Dewhurst, and W. Knight. 2010. *Product Design for Manufacture and Assembly.* 3rd Ed. Boca Raton, FL, USA: CRC Press.

Bralla, J. 1998. *Design for Manufacturability Handbook.* New York, NY, USA: McGraw Hill Professional.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Affordability

A system is affordable to the degree that system performance, cost, and schedule constraints are balanced over the system life, while mission needs are satisfied in concert with strategic investment and organizational needs (INCOSE 2011). Design for affordability is the practice of considering affordability as a design characteristic or constraint.

Increasing competitive pressures and the scarcity of resources demand that systems engineering (SE) improve affordability. Several recent initiatives have made affordability their top technical priority. They also call for a high priority to be placed on research into techniques — namely, improved systems autonomy and human performance augmentation — that promise to reduce labor costs, provide more efficient equipment to reduce supply costs, and create adaptable systems whose useful lifetime is extended cost-effectively.

Yet methods for cost and schedule estimation have not changed significantly to address these new challenges and opportunities. There is a clear need for

- new methods to analyze tradeoffs between cost, schedule, effectiveness, and resilience;
- new methods to adjust priorities and deliverables to meet budgets and schedules; and
- more affordable systems development processes.

All of this must be accomplished the context of the rapid changes underway in technology, competition, operational concepts, and workforce characteristics.

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Historically, cost and schedule estimation has been decoupled from technical SE tradeoff analyses and decision reviews. Most models and tools focus on evaluating either cost-schedule performance or technical performance, but not the tradeoffs between the two. Meanwhile, organizations and their systems engineers often focus on affordability to minimize acquisition costs. They are then drawn into the easiest-first approaches that yield early successes, at the price of being stuck with brittle, expensive-to-change architectures that increase technical debt and life cycle costs.

Two indications that the need for change is being recognized in systems engineering are that the *INCOSE SE Handbook* now includes affordability as one of the criteria for evaluating requirements (INCOSE 2011); and, there is a trend in SE towards stronger focus on maintainability, flexibility, and evolution (Blanchard, Verma, and Peterson 1995).

There are pitfalls for the unwary. Autonomous systems experience several hazardous failure modes, including

- **system instability due to positive feedback —** where an agent senses a parameter reaching a control limit and gives the system a strong push in the other direction, causing the system to rapidly approach the other control limit, causing the agent (or another) to give it an even stronger push in the original direction, and so on
- **self-modifying autonomous agents which fail** after several self-modifications — the failures are difficult to debug because the agent's state has been changing
- **autonomous agents performing weakly at commonsense reasoning** about system control decisions by human operators, and so tend to reach wrong conclusions and make wrong decisions about controlling the system
- **multiple agents making contradictory decisions** about controlling the system, and lacking the ability to understand the contradiction or to negotiate a solution to resolve it

Modularization of the system's architecture around its most frequent sources of change (Parnas 1979) is a key SE principle for affordability. This is because when changes are needed, their side effects are contained in a single systems element, rather than rippling across the entire system.

This approach creates the need for three further improvements:

- refocusing the system requirements, not only on a snapshot of current needs, but also on the most likely sources of requirements change, or evolution requirements;
- monitoring and acquiring knowledge about the most frequent sources of change to better identify requirements for evolution; and
- evaluating the system's proposed architecture to access how well it supports the evolution requirements, as well as the initial snapshot requirements.

This approach can be extended to produce several new practices. Systems engineers can

- identify the commonalities and variability across the families of products or product lines, and develop architectures for creating (and evolving) the common elements *once* with plug-compatible interfaces for inserting the variable elements (Boehm, Lane, and Madachy 2010);
- extrapolate principles for service-oriented system elements that are characterized by their inputs, outputs, and assumptions, and that can easily be composed into systems in which the sources of change were not anticipated; and
- develop classes of smart or autonomous systems whose many sensors identify needed changes, and whose autonomous agents determine and effect those changes in microseconds, or much more rapidly than humans can, reducing not only reaction time, but also the amount of human labor needed to operate the systems, thus improving affordability.

System Description

Information to be supplied at a later date.

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Information to be supplied at a later date.

Discipline Standards

Information to be supplied at a later date.

Personnel Considerations

Autonomous systems need human supervision, and the humans involved require better methods for trend analysis and visualization of trends (especially, undesired ones).

There is also the need, with autonomous systems, to extend the focus from life cycle costs to total ownership costs, which encompass the costs of failures, including losses in sales, profits, mission effectiveness, or human quality of life. This creates a further need to evaluate affordability in light of the value added by the system under consideration. In principle, this involves evaluating the system’s total cost of ownership with respect to its mission effectiveness and resilience across a number of operational scenarios. However, determining the appropriate scenarios and their relative importance is not easy, particularly for multi-mission systems of systems. Often, the best that can be done involves a mix of scenario evaluation and evaluation of general system attributes, such as cost, schedule, performance, and so on.

As for these system attributes, different success-critical stakeholders will have different preferences, or utility functions, for a given attribute. This makes converging on a mutually satisfactory choice among the candidate system solutions a difficult challenge involving the resolution of the multi-criteria decision analysis (MCDA) problem among the stakeholders (Boehm and Jain 2006). This is a well-known problem with several paradoxes, such as Arrow’s impossibility theorem that describes the inability to guarantee a mutually optimal solution among several stakeholders, and several paradoxes in stakeholder preference aggregation in which different voting procedures produce different winning solutions. Still, groups of stakeholders need to make decisions, and various negotiation support systems enable people to better understand each other’s utility functions and to arrive at mutually satisfactory decisions, in which no one gets everything that they want, but everyone is at least as well off as they are with the current system.

Also see System Analysis for considerations of cost and affordability in the technical design space.

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be supplied at a later date.

Practical Considerations

Pitfalls

Information to be provided at a later date.

Proven Practices

Information to be provided at a later date.

Other Considerations

Information to be provided at a later date.

Primary References

Works Cited

Blanchard, B., D. Verma, and E. Peterson. 1995. *Maintainability: A Key to Effective Serviceability and Maintenance Management.* New York, NY, USA: Wiley and Sons.

Boehm, B., J. Lane, and R. Madachy. 2010. "Valuing System Flexibility via Total Ownership Cost Analysis." Proceedings of the NDIA SE Conference, October 2010, San Diego, CA, USA.

Boehm, B., and A. Jain. 2006. "A Value-Based Theory of Systems Engineering." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium (IS), July 9-13, 2006, Orlando, FL, USA.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2. p. 79.

Parnas, D.L. 1979. "Designing Software for Ease of Extension and Contraction." *IEEE Transactions on Software Engineering*. 5 (2): 128-138.

Primary References

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2. p. 79.

Blanchard, B., D. Verma, and E. Peterson. 1995. *Maintainability: A Key to Effective Serviceability and Maintenance Management.* New York, NY, USA: Wiley and Sons.

Parnas, D.L. 1979. "Designing Software for Ease of Extension and Contraction." *IEEE Transactions on Software Engineering*. 5 (2): 128-138.

Additional References

Kobren, Bill. 2011. "Supportability as an Affordability Enabler: A Critical Fourth Element of Acquisition Success Across the System Life Cycle." *Defense AT&L: Better Buying Power.* Accessed August 28, 2012. Available: http://www.dau.mil/pubscats/ATL%20Docs/Sep-Oct11/Kobren.pdf.

Myers, S.E., P.P. Pandolfini, J.F. Keane, O. Younossi, J.K. Roth, M.J. Clark, D.A. Lehman, and J.A. Dechoretz. 2000. "Evaluating affordability initiatives." *Johns Hopkins APL Tech. Dig.* 21 (3): 426–437.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Environmental Engineering

Environmental engineering addresses four issues that arise in system design and operation. They include: (1) design for a given operating environment, (2) environmental impact, (3) green design, and (4) compliance with environment regulations.

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

A system is designed for a particular operating environment. Product systems, in particular, routinely consider conditions of temperature and humidity. Depending on the product, other environmental conditions may need to be considered, including UV exposure, radiation, magnetic forces, vibration, and others. The allowable range of these conditions must be specified in the requirements for the system.

Requirements

The general principles for writing requirements also apply to specifying the operating environment for a system and its elements. Requirements are often written to require compliance with a set of standards.

System Description

Information to be supplied at a later date.

Discipline Management

Many countries require assessment of environmental impact of large projects before regulatory approval is given. The assessment is documented in an environmental impact statement (EIS). In the United States, a complex project can require an EIS that greatly adds to the cost, schedule, and risk of the project.

Scope

In the U.S., the process in Figure 1 is followed. A proposal is prepared prior to a project being funded. The regulator examines the proposal. If it falls into an excluded category, no further action is taken. If not, an environmental assessment is made. If that assessment determines a finding of no significant impact (FONSI), no further action is taken. In all other cases, an environmental impact statement is required.

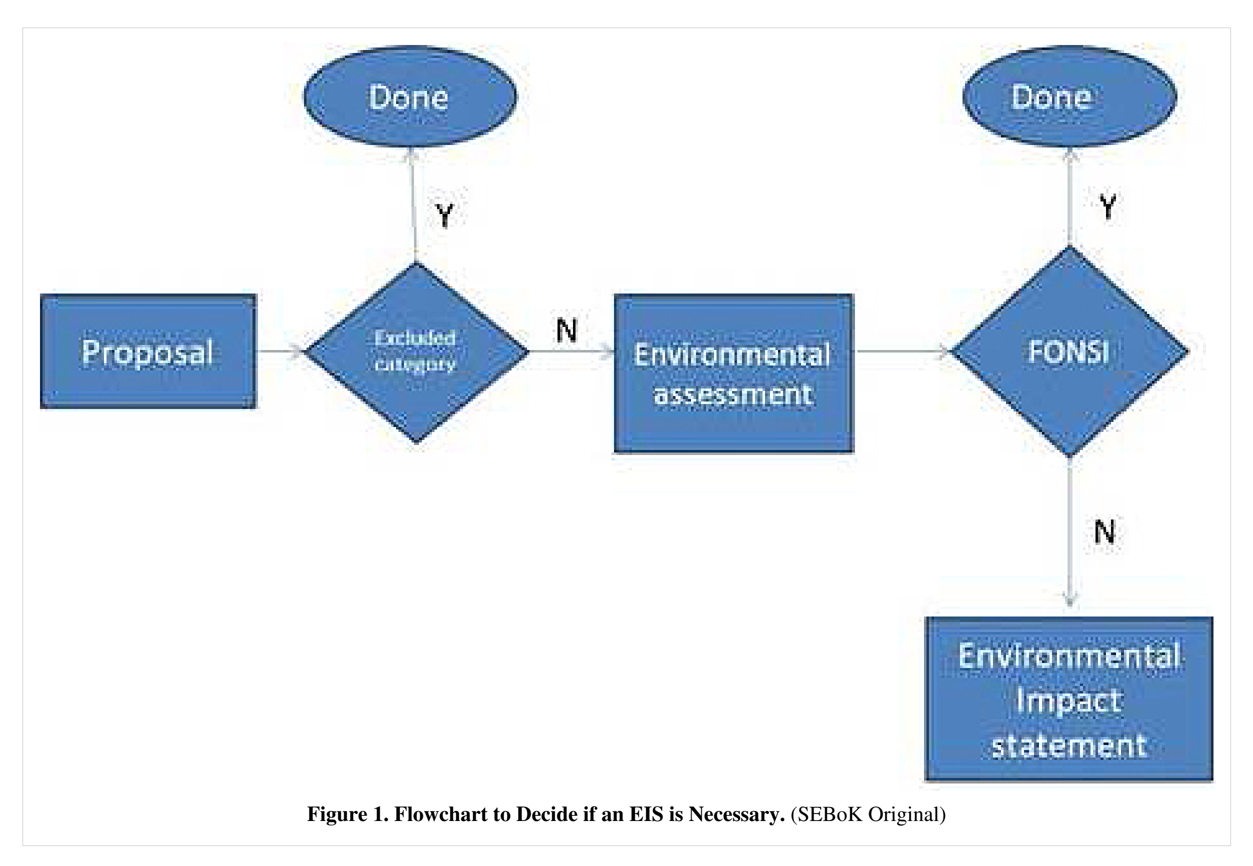


Figure 1. Flowchart to Decide if an EIS is Necessary. (SEBoK Original)

Preparation of an EIS is a resource significant task. Bregman (2000) and Kreske (1996) provide accessible overviews of the process. Lee and Lin (2000) provide a handbook of environmental engineering calculations to aid in the technical submission. Numerous firms offer consulting services.

Legal References

Basic references in the U.S. include the National Environmental Policy Act of 1969 and its implementing regulations (NEPA 1969) and the European commission directive (EC 1985). State and local regulations can be extensive; Burby and Paterson (1993) discuss improving compliance.

Cost and Schedule Implications

Depending on the scale of the project, the preparation of an EIS can take years and cost millions. For example, the EIS for the Honolulu light rail project took four years and cost $156M (Hill 2011). While a project may proceed even if the EIS finds a negative impact, opponents to a project may use the EIS process to delay a project. A common tactic is to claim the EIS was not complete in that it omitted some environmental impacts. Eccleston (2000) provides a guide to planning for EIS.

Energy Efficiency

There is a large amount of literature that has been published about design for energy efficiency. Lovins (2010) offer ten design principles. He also provides case studies (Lovins et al. 2011). Intel (2011) provides guidance for improving the energy efficiency of its computer chips. A great deal of information is also available in regard to the efficient design of structures; DOE (2011) provides a good overview.

Increased energy efficiency can significantly reduce total life cycle cost for a system. For example, the Toyota Prius was found to have the lowest life cycle cost for 60,000 miles, three years despite having a higher initial purchase price (Brown 2011).

Carbon Footprint

Increased attention is being paid to the emission of carbon dioxide. BSI British Standards offers a specification for assessing life cycle greenhouse emissions for goods and services (BSI 2011).

Sustainability

Graedel and Allenby (2009), Maydl (2004), Stasinopoulos (2009), Meryman (2004), and Lockton and Harrison (2008) discuss design for sustainability. Sustainability is often discussed in the context of the UN report on Our Common Future (WCED 1987) and the Rio Declaration (UN 1992).

Discipline Relationships

An enterprise must attend to compliance with the various environmental regulations. Dechant et al. (1994) provide the example of a company in which 17% of every sales dollar goes toward compliance activities. They discuss gaining a competitive advantage through better compliance. Gupta (1995) studies how compliance can improve the operations function. Berry (1998) and Nash (2001) discuss methods for environmental management by the enterprise.

Interactions

Information to be supplied at a later date.

Dependencies

ISO14001 sets the standards for organization to comply with environmental regulations. Kwon and Seo (2002) discuss this in a Korean context, and Whitelaw (2004) presents a handbook on implementing ISO14001.

Discipline Standards

Depending on the product being developed, standards may exist for operating conditions. For example, ISO 9241-6 specifies the office environment for a video display terminal. Military equipment may be required to meet MILSTD 810G standard (DoD 2014) in the US, or DEF STAN 00-35 in the UK (MoD 2006).

The U.S. Federal Aviation Administration publishes a list of EIS best practices (FAA 2002).

The U.S. Environmental Protection Agency (EPA) defines Green Engineering (glossary) as: the design, commercialization, and use of processes and products, which are feasible and economical, while minimizing (1) generation of pollution at the source and (2) risk to human health and the environment (EPA 2011). Green engineering embraces the concept that decisions to protect human health and the environment can have the greatest impact and cost effectiveness when applied early to the design and development phase of a process or product.

The EPA (2011) offers the following principles of green engineering:

- Engineer processes and products holistically, use systems analysis, and integrate environmental impact assessment tools.
- Conserve and improve natural ecosystems while protecting human health and well-being.
- Use life-cycle thinking in all engineering activities.
- Ensure that all material and energy inputs and outputs are as inherently safe and benign as possible.
- Minimize depletion of natural resources.
- Strive to prevent waste.
- Develop and apply engineering solutions, while being cognizant of local geography, aspirations, and cultures.
- Create engineering solutions beyond current or dominant technologies; additionally, improve, innovate, and invent (technologies) to achieve sustainability.
- Actively engage communities and stakeholders in development of engineering solutions.

Personnel Considerations

Information to be supplied at a later date.

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be supplied at a later date.

Practical Considerations

Pitfalls

Information to be provided at a later date.

Proven Practices

Information to be provided at a later date.

Other Considerations

Information to be provided at a later date.

References

Works Cited

Berry, MA. 1998. "Proactive corporate environmental management: a new industrial revolution." *The Academy of Management Executive*, 12 (2): 38-50.

Bregman, J.I. 2000. *Environmental Impact Statements*, 2nd ed. Boca Raton, FL, USA: CRC Press.

Brown, C. 2011 "The Green Fleet Price Tag." Business Fleet. Available: http://www.businessfleet.com/Article/Story/2011/07/The-Green-Fleet-Price-Tag.aspx.

BSI. 2011. "Specification for the assessment of the life cycle greenhouse gas emissions of goods and service, PAS 2050:2011." London, UK: British Standards Institution (BSI). Available: http://shop.bsigroup.com/en/forms/PASs/PAS-2050.

Burby, R.J., and R.G. Paterson. 1993. "Improving compliance with state environmental regulations." *Journal of Policy Analysis and Management*, 12(4): 753–772.

Dechant, K., B. Altman, R.M. Downing, and T. Keeney. 1994. "Environmental Leadership: From Compliance to Competitive Advantage." *Academy of Management Executive*, 8(3): 7.

DoD. 2014. *Department of Defense Test Method Standard: Environmental Engineering Considerations and Laboratory Tests*, MIL-STD-810G Change Notice 1. Washington, DC, USA: US Army Test and Evaluation Command, US Department of Defense (DoD). Accessed November 4, 2014. Available: http://www.atec.army.mil/publications/Mil-Std-810G/MIL-STD-810G%20CN1.pdf.

Eccleston, C. 2000. *Environmental Impact Statements: A Comprehensive Guide to Project and Strategic Planning*. New York, NY, USA: Wiley.

EPA. 2011. "Green Engineering. Environmental Protection Agency (EPA)." Available: http://www.epa.gov/oppt/greenengineering/.

EC. 1985. "Council Directive of 27 June 1985 on the assessment of the effects of certain public and private projects on the environment (85/337/EEC)." European Commission (EC). Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1985L0337:20090625:EN:PDF.

FAA. 2002. "Best Practices for Environmental Impact Statement (EIS) Management." Federal Aviation Administration (FAA). Available: http://www.faa.gov/airports/environmental/eis_best_practices/?sect=intro.

Graedel, T.E., and B.R. Allenby. 2009. *Industrial Ecology and Sustainable Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Gupta, M.C. 1995. "Environmental management and its impact on the operations function." *International Journal of Operations and Production Management*, 15 (8): 34-51.

Hill, T. 2011. "Honolulu Rail's Next Stop?" *Honolulu Magazine*. July 2011.

Intel. 2011. "Energy Efficiency." Intel Corporation. Accessed: August 29, 2012. Available: http://www.intel.com/intel/other/ehs/product_ecology/energy.htm.

Kreske, D.L. 1996. *Environmental impact statements: a practical guide for agencies, citizens, and consultants*. New York, NY: Wiley.

Kwon, D.M., and M.S. Seo. 2002. "A study of compliance with environmental regulations of ISO 14001 certified companies in Korea." *Journal of Environmental Management*. 65 (4): 347-353.

Lee, C.C., and S.D. Lin. 2000. *Handbook of Environmental Engineering Calculations*. New York, NY, USA: McGraw Hill Professional.

Lockton, D., and D. Harrison. 2008. "Making the user more efficient: Design for sustainable behaviour." *International Journal of Sustainable Engineering*. 1 (1): 3-8.

Lovins, A. 2010. "Factor Ten Engineering Design Principles," version 1.0. Available: http://www.rmi.org/Knowledge-Center/Library/2010-10_10xEPrinciples.

Lovins, A., et al. 2011. "Case Studies." Available: http://move.rmi.org/markets-in-motion/case-studies/.

Maydl, Peter. 2004. "Sustainable Engineering: State-of-the-Art and Prospects." *Structural Engineering International*. 14 (3): 176-180.

Meryman, H. 2004. "Sustainable Engineering Using Specifications to Make it Happen." *Structural Engineering International*. 14 (3).

MoD. 2006. *Standard 00-35, Environmental Handbook for Defence Materiel (Part 3) Environmental Test Methods*. London, England, UK: UK Ministry of Defence (MoD). Available: http://www.everyspec.com/DEF-STAN/download.php?spec=DEFSTAN00-35_I4.029214.pdf.

Nash, J. 2001. *Regulating from the inside: can environmental management systems achieve policy goals?* Washington, DC, USA: Resources for the Future Press.

NEPA. 1969. *42 USC 4321-4347. National Environmental Policy Act (NEPA)*. Accessed January 15, 2012. Available: http://ceq.hss.doe.gov/nepa/regs/nepa/nepaeqia.htm.

Stasinopoulos, P. 2009. *Whole system design: an integrated approach to sustainable engineering*. London, UK: Routledge.

UN. 1992. "Rio Declaration on Environment and Development." United Nations (UN). Available: http://www.unep.org/Documents.Multilingual/Default.asp?documentid=78&articleid=1163.

Whitelaw, K. 2004. *ISO 14001: Environmental Systems Handbook*, 2nd ed. Oxford, UK: Elsevier.

WCED. 1987. "Our Common Future. World Commission on Economic Development (WCED)." Available: http://www.un-documents.net/wced-ocf.htm.

Primary References

Bregman, J.I. 2000. *Environmental Impact Statements*, 2nd ed. Boca Raton, FL, USA: CRC Press.

Graedel, T.E., and B.R. Allenby. 2009. *Industrial Ecology and Sustainable Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Lee, C.C., and S.D. Lin. 2000. *Handbook of Environmental Engineering Calculations*. New York, NY, USA: McGraw Hill Professional.

Whitelaw, K. 2004. *ISO 14001: Environmental Systems Handbook*, 2nd ed. Oxford, UK: Elsevier.

Additional References

None.

< Previous Article | Parent Article | Next Article (Part 7) >

SEBoK v. 1.9.1, released 16 October 2018

Part 7: Systems Engineering Implementation Examples

Systems Engineering Implementation Examples

Part 7 is a collection of systems engineering (SE) implementation examples to illustrate the principles described in the Systems Engineering Body of Knowledge (SEBoK) Parts 1-6. These examples describe the application of SE practices, principles, and concepts in real settings.

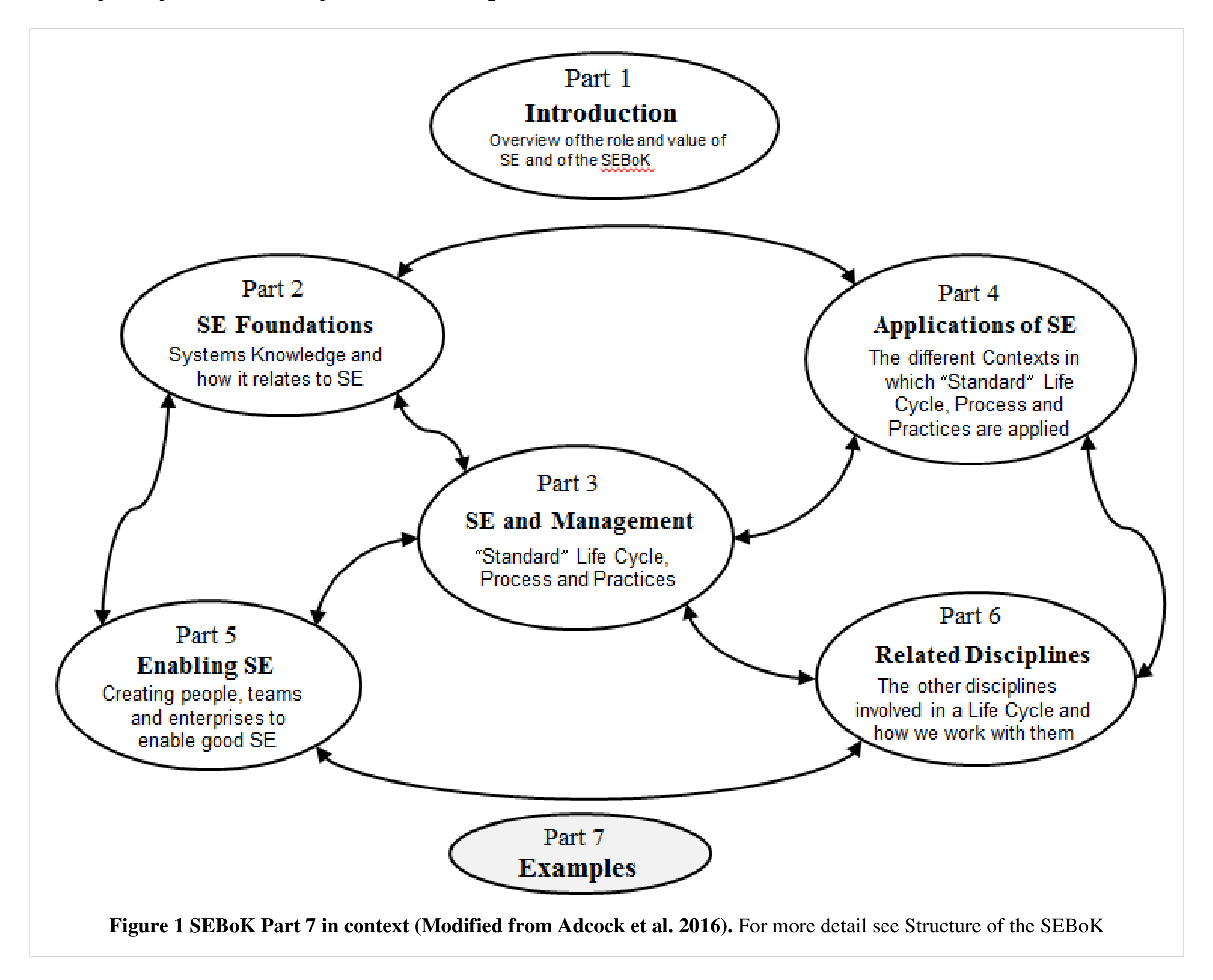


Figure 1 SEBoK Part 7 in context (Modified from Adcock et al. 2016). For more detail see Structure of the SEBoK

The intent is to provide typical instances of the application of systems engineering (SE) and relate these to key SE principles and concepts from the rest of the SEBoK. This can improve the practice of SE by illustrating to students, educators, and practitioners the benefits of effective practice, as well as the risks and liabilities of poor practice.

A published case study will typically describe aspects of the practice of SE in a particular situation and then provide comments and critique of that practice. Where possible, examples in the SEBoK refer to published case studies and relate the discussions in them to appropriate areas of the SEBoK. In some case, good or bad examples of SE practice are available but have not been documented in a case study. In these cases the SEBoK authors have described and commented on these examples directly.

A matrix of implementation examples is used to map these examples to main topics in the SEBoK which they cover.

More examples will be added over time to highlight the different aspects and applications of SE. In addition, new examples can be added to demonstrate the evolving state of practice, such as the application of model-based SE and the engineering of complex, adaptive systems.

Knowledge Areas in Part 7

Part 7 is organized in the following way:

- Matrix of Implementation Examples
- Implementation Examples

Value of Implementation Examples

Learning from critical examples has been used for decades in medicine, law, and business to help students learn fundamentals and to help practitioners improve their practice. A Matrix of Implementation Examples is used to show the alignment of systems engineering case studies to specific areas of the SEBoK. This matrix is intended to provide linkages between each implementation example to the discussion of the systems engineering principles illustrated. The selection of examples covers a variety of sources, domains, and geographic locations. Both effective and ineffective use of systems engineering principles are illustrated.

The United States Air Force Center for Systems Engineering (AF CSE) has developed a set of case studies "to facilitate learning by emphasizing the long-term consequences of the systems engineering/programmatic decisions on cost, schedule, and operational effectiveness." (USAF Center for Systems Engineering 2011) The AF CSE is using these cases to enhance SE curriculum. The cases are structured using the Friedman-Sage framework (Friedman and Sage 2003; Friedman and Sage 2004, 84-96), which decomposes a case into contractor, government, and shared responsibilities in the following nine concept areas:

1. Requirements Definition and Management
2. Systems Architecture Development
3. System/Subsystem Design
4. Verification/Validation
5. Risk Management
6. Systems Integration and Interfaces
7. Life Cycle Support
8. Deployment and Post Deployment
9. System and Program Management

This framework forms the basis of the case study analysis carried out by the AF CSE. Two of these case studies are highlighted in this SEBoK section, the Hubble Space Telescope Case Study and the Global Positioning System Case Study.

The United States National Aeronautics and Space Administration (NASA) has a catalog of more than fifty NASA-related case studies (NASA 2011). These case studies include insights about both program management and systems engineering. Varying in the level of detail, topics addressed, and source organization, these case studies are used to enhance learning at workshops, training, retreats, and conferences. The use of case studies is viewed as important by NASA since "organizational learning takes place when knowledge is shared in usable ways among organizational members. Knowledge is most usable when it is contextual" (NASA 2011). Case study teaching is a method for sharing contextual knowledge to enable reapplication of lessons learned. The MSTI Case Study is from this catalog.

References

Works Cited

Adcock, R., Hutchison, N., Nielsen, C., 2016, "Defining an architecture for the Systems Engineering Body of Knowledge," Annual IEEE Systems Conference (SysCon) 2016.

Friedman, G.R., and A.P. Sage. 2003. *Systems Engineering Concepts: Illustration Through Case Studies.*

Friedman, G.R., and A.P. Sage. 2004. "Case Studies of Systems Engineering and Management in Systems Acquisition." *Systems Engineering.* 7 (1): 84-96.

NASA. 2011. *A Catalog of NASA-Related Case Studies.* Goddard Space Flight Center: Office of the Chief Knowledge Officer, National Aeronautics and Space Administration (NASA). Updated June 2011. Accessed September 2011. Available: http:/ / www. nasa. gov/ centers/ goddard/ pdf/ 450420main_NASA_Case_Study_Catalog.pdf.

United States Air Force (USAF) Center for Systems Engineering. 2011. *Why Case Studies?*. Wright-Patterson Air Force Base, Ohio, USA: Air Force Institute of Technology (AFIT), US Air Force. Accessed September 2011. Available: http://www.afit.edu/cse/cases.cfm.

Primary References

Friedman, G., and A.P. Sage. 2004. "Case Studies of Systems Engineering and Management in Systems Acquisition". *Systems Engineering* 7(1): 84-96.

Gorod, A., B.E. White, V. Ireland, S.J. Gandhi, and B.J. Sauser. 2014. *Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering*. Boca Raton, FL: CRC Press, Taylor & Francis Group.

NASA. A Catalog of NASA-Related Case Studies. Greenbelt, MD, USA: Office of the Chief Knowledge Officer, Goddard Space Flight Center, National Aeronautics and Space Administration (NASA). Updated June 2011. Accessed December 5 2014 at NASA http:/ / www. nasa. gov/ centers/ goddard/ pdf/ 450420main_NASA_Case_Study_Catalog.pdf.

United States Air Force (USAF) Center for Systems Engineering. 2011. Why Case Studies?. Wright-Patterson Air Force Base, OH, USA: Air Force Institute of Technology (AFIT).

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Matrix of Implementation Examples

The following matrix maps the Systems Engineering Implementation Examples to topics in the Systems Engineering Body of Knowledge (SEBoK). It provides both a list of systems engineering implementation examples for topics of interest, and a list of relevant topics for each implementation example. Since the number of topics in the SEBoK is extensive, only a subset are included here for clarity. For additional information, see the example of interest and the corresponding SEBoK topic.

Organization and Mapping of Examples to the SEBoK

The following short titles shown in Table 1 are used for the implementation examples:

Table 1. Short Titles for the SEBoK Examples. (SEBoK Original)

Case Studies	
BT	Business Transformation
ATC	NextGen Air Traffic Control
NASA	NASA's Mission to Saturn
HST	Hubble Space Telescope
GPS	Global Positioning System
GPS II	Global Positioning System II
Radiation	Medical Radiation
FBI VCF	FBI Virtual Case File System
MSTI	Miniature Seeker Technology Integration
Infusion Pump	Next Generation Medical Infusion Pump
DfM	Design for Maintainability
CAS	Complex Adaptive Operating System
PM	Project Management
TS	Taxi Service
TS	Taxi Service
SWFTS	SWFTS MBSE
Bag Handling	Denver Airport Baggage Handling System
VA Sub	Virginia Class Submarine
Route Mod	UK West Coast Route Modernisation Project
Water Mgmt	Singapore Water Management
FAA AAS	FAA Advanced Automation System
Light Rail	Standard Korean Light Transit System
TMT	Thirty-Meter Telescope

Table 2 shows how the topics (each row) align with the first set of implementation examples (each column):

Table 2. Implementation Examples based on published case studies. (SEBoK Original)

SEBoK Topic	BT	ATC	NASA	HST	GPS	GPS II	Radiation	FBI VCF	MSTI	Infusion Pump	DfM	CAS	PM	TS	SWFTS	NHS
Systems Thinking					X	X		X	X	X	X	X	X	X		
Models and Simulation		X	X			X							X	X	X	
Product Systems Engineering				X	X		X	X								X
Service Systems Engineering			X			X								X		X
Enterprise Systems Engineering	X	X	X			X							X	X		X
Systems of Systems (SoS)	X	X				X								X		X
Life Cycle Models	X			X	X				X	X	X	X			X	
Business or Mission Analysis	X	X				X		X		X			X	X		
Stakeholder Needs and Requirements	X	X		X		X		X		X			X	X	X	
System Requirements				X	X			X		X			X	X	X	
System Architecture	X	X			X	X			X	X						
System Analysis			X		X							X	X	X		
System Implementation					X						X			X		
System Integration	X		X	X	X	X			X		X		X	X		
System Verification			X		X		X		X	X						
System Validation	X				X	X	X		X	X						
System Deployment					X	X					X					
Operation of the System					X	X				X	X			X		X
System Maintenance																X
Logistics																
Planning	X	X		X				X		X			X			
Assessment and Control							X	X	X				X	X		
Risk Management			X	X	X	X	X	X		X		X	X			
Measurement									X							
Decision Management	X								X			X	X	X		
Configuration Management					X	X			X		X	X				
Information Management			X									X	X	X		
Quality Management															X	

Enabling Systems Engineering	X	X	X			X	X	X	X	X	X
Related Disciplines			X	X	X		X		X		

Table 3 shows how the topics (each row) align with the second set of implementation examples (each column):

Table 3. Implementation Examples Developed for SEBoK. (SEBoK Original)

SEBoK Topic (Part 3)	Bag Handling	VA Sub	Route Mod	Water Mgmt	FAA AAS	Light Rail
Business or Mission Analysis			X	X		X
Stakeholder Needs and Requirements			X	X		X
System Requirements			X		X	X
System Architecture	X	X		X	X	X
System Analysis	X			X	X	X
System Implementation						X
System Integration	X			X		X
System Verification	X				X	X
System Validation	X				X	X
System Deployment						X
Operation of the System				X		X
System Maintenance						X
Logistics	X					
Planning	X		X	X	X	X
Assessment and Control					X	X
Risk Management	X		X	X	X	X
Measurement						X
Decision Management	X		X	X		
Configuration Management	X		X		X	
Information Management			X			
Quality Management				X		

References

Works Cited

None.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Implementation Examples

Characteristics of the organization(s) which do SE have an impact on the engineering itself. It is for this reason that the examples are grouped into three categories: government examples, commercial examples, and examples which represent both issues from both ("combined"). As the SEBoK continues to evolve, additional categorizations may be developed. A matrix is used to map the specific systems engineering principles illustrated in each example to the associated SEBoK topics.

SEBoK Examples

- Commercial Examples
  - Complex Adaptive Operating System: Project Management
  - Complex Adaptive Project Management System
  - Complex Adaptive Taxi Service Scheduler
  - Denver Airport Baggage Handling System
  - Global Positioning System
  - Global Positioning System II
  - Next Generation Medical Infusion Pump
  - Medical Radiation
  - Successful Business Transformation within a Russian Information Technology Company
- Government Examples
  - FBI Virtual Case File System
  - Design for Maintainability
  - FAA Advanced Automation System (AAS)
  - Federal Aviation Administration Next Generation Air Transportation System
  - How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn
  - Hubble Space Telescope
  - Northwest Hydro System
  - Singapore Water Management
  - Submarine Warfare Federated Tactical Systems
  - UK West Coast Route Modernisation Project
  - Virginia Class Submarine

- Combined Examples
  - Applying a Model-Based Approach to Support Requirements Analysis on the Thirty-Meter Telescope
- Miniature Seeker Technology Integration Spacecraft
- Standard Korean Light Transit System

Characterization of Examples

Systems engineering (SE) principles described in the SEBoK Parts 1-6 are illustrated in Part 7, Systems Engineering Implementation Examples. These examples describe the application of systems engineering practices, principles, and concepts in real settings and can be used to improve the practice of systems engineering by illustrating to students, practitioners, and those new to SE the benefits of effective practice and the risks of poor practice.

The examples here have been developed by examining previously published case studies from external sources that demonstrate the real world examples of systems engineering principles were examined and summarized by the BKCASE team. These summaries include links to the original documentation as well as links to the relevant areas of the SEBoK highlighted in the example.

Systems engineering (SE) examples can be characterized in terms of at least two relevant parameters, viz., their degrees of complexity and engineering difficulty, for example. Although a so-called quad chart is likely an oversimplification, a 2 x 2 array can be used to make a first-order characterization, as shown in Figure 1.

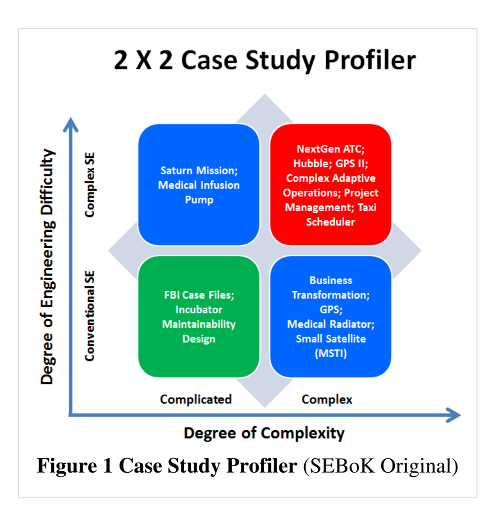


Figure 1 Case Study Profiler (SEBoK Original)

The x-axis depicts complicated, the simplest form of complexity, at the low-end on the left, and complex, representing the range of all higher forms of complexity on the right.The y-axis suggests how difficult it might be to engineer (or re-engineer) the system to be improved, using Conventional (classical or traditional) SE, at the low-end on the bottom, and Complex SE, representing all more sophisticated forms SE, on the top. This upper range is intended to cover system of systems (SoS) engineering (SoSE), enterprise systems engineering (ESE), as well as Complex SE (CSE).The distinctions among these various forms of SE may be explored by visiting other sections of the SEBoK. In summary, the SEBoK case study editors have placed each case study in one of these four quadrants to provide readers with a suggested characterization of their case study's complexity and difficulty. For sake of compactness the following abbreviations have been used:

- Business Transformation (Successful Business Transformation within a Russian Information Technology Company)
- NextGen ATC (Federal Aviation Administration Next Generation Air Transportation System)
- Saturn Mission (How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn)
- Hubble (Hubble Space Telescope Case Study)
- GPS and GPS II (Global Positioning System Case Study)

- Medical Radiator (Medical Radiation Case Study)
- FBI Case Files (FBI Virtual Case File System Case Study)
- Small Satellite MSTI (MSTI Case Study)
- Medical Infusion Pump (Next Generation Medical Infusion Pump Case Study)
- Incubator Maintainability Design (Design for Maintainability)
- Complex Adaptive Operations (Complex Adaptive Operating System)
- Taxi Scheduler (The Development of the First Real-Time Complex Adaptive Scheduler for a London Taxi Service)
- Project Management (The Development of a Real-Time Complex Adaptive Project Management Systems)
- SWFTS MBSE (Submarine Warfare Federated Tactical Systems Case Study)

References

Works Cited

None.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Commercial Examples

Complex Adaptive Taxi Service Scheduler

This article is based around a **London Taxi Service Case Study** (Rzevski and Skobelev, 2014). The case study focuses on the development of a Real-Time Complex Adaptive Scheduler for a London Taxi Service capable of managing the complexity of many hundreds of taxi journeys in an unpredictable and changing environment, while fitting into the goals and values of the Enterprise.

Background

When this project was initiated, our client, the largest and the best-known minicab (taxi) operator in London had a fleet of more than 2,000 vehicles, each with a Global Positioning System (GPS) navigation system. The fleet comprised a variety of vehicles, including minivans and Sport Utility Vehicles (SUVs), some with equipment to match special customer requirements. Typically approximately 700 drivers worked concurrently, competing with each other for customers.

The company had a modern Enterprise Resource Planning (ERP) system and a call centre with over 100 operators receiving orders concurrently. Some orders were received through the company website. A large team of skilled dispatchers allocated vehicles to customers.

Main characteristics of the taxi service were as follows:

- More than 13,000 orders per day
- Occasionally more than 1,500 orders per hour (1 order every 2.4 seconds)
- Unpredictable order arrival times and locations
- Various clients, e.g., personal, corporate, Very Important Persons (VIPs), a variety of discounted tariffs, special requirements suitable for the disabled, small children (child seats), transportation of pets, etc.
- Many freelance drivers who leased cars from the company and were allowed to start and finish their shifts at times that suited them, which may have differed from day to day
- Clients in central London were guaranteed pick up times within 15 minutes of order placement
- Fundamentally the company tried to find the best economic match of vehicle to every client
- However, dynamic exceptions to this basic requirement included: matching drivers going to and from home with passengers travelling in the same direction (to reduce drivers' idle runs); and giving priority to drivers with less work during a particular day (to increase drivers' satisfaction with working conditions)

No pre-planned taxi schedule was viable because any of the following unpredictable "Disruptive Events" occurred every 2 to 10 seconds:

- Order arrival, change, or cancellation
- Changes in driver profile, status, or location
- Client no-show
- Vehicle failure
- Delays due to traffic congestion, or queues at airports, railway stations, etc.

Purpose

Rescheduling up to 700 independent entities, travelling in London under unpredictable conditions that change every few seconds represented an exceedingly complex task, which was not feasible to accomplish using any known mathematical method.

Manual scheduling, as practiced, could not handle the frequent disruptive events. Many perturbations, such as unexpected delays, had to be ignored by the human dispatchers.

Therefore the project's objective was to provide effective, real-time, automated assistance to accommodate the disruptions that drove the scheduling. Thus, the project purpose became the development of a complex adaptive software system capable of managing the taxi operation complexity described above with the aim of substantially improving: (1) operational profitability; (2) customer service quality; and (3) driver working conditions.

The planned transformation was from a manual to semi-automated managed taxi operation that facilitated optional human dispatcher interactions with a complex adaptive system scheduler. A thorough analysis of contemporary practices showed that such a transformation has never been achieved before. To the best of the team knowledge, there were no real-time schedulers of taxi operations in existence anywhere in the world.

Challenges

The team undertaking the development of a new real-time scheduler for this client had vast experience of designing and implementing complex adaptive software, and therefore no particular challenges were anticipated. The multi-agent technology, which underpinned the system, was well understood by the team, and a methodology for managing complexity (Rzevski and Skobelev, 2014) of the task was in place.

Systems Engineering Practices

Overview

The complexity of the taxi service ruled out all conventional systems engineering practices. The real-time adaptive scheduler for the client's taxi service was developed using multi-agent software technology.

The scheduler design consisted of the following major components (Rzevski and Skobelev, 2014):

1. Knowledge Base containing domain useful information relevant to the client's taxi service
2. Multi-agent Virtual World which models the Real World of the taxi service and is capable of managing its complexity
3. Communication channels between the Virtual and Real Worlds which enable the Virtual World management of the Real World with or without human intervention.

The system was designed to behave as follows. In reaction to every disruptive event, Order Agents, assigned to every received order, and Driver Agents, assigned to every working driver, negotiate, through the exchange of messages, the most suitable Order-Driver match. Before starting negotiations these software agents consult the Knowledge Base for the current negotiation rules. Once the best possible match (under prevailing circumstances) is agreed, the result is communicated to Drivers, who are free to accept or reject the task (Glaschenko et al. 2009). All this is depicted simply in the figure below.

After a successful prototype implementation, a basic version of the complex adaptive scheduler was developed as described below.

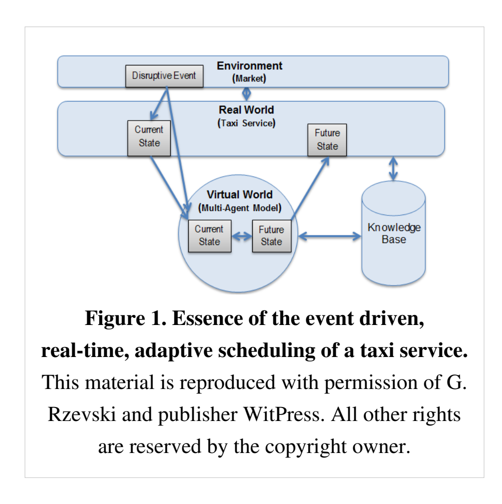


Figure 1. Essence of the event driven, real-time, adaptive scheduling of a taxi service.

Knowledge Base

The Knowledge Base consisted of: (1) Ontology, containing conceptual knowledge as a semantic network; and (2) Values, in standard databases.

The basic Ontology contained two Object Classes: Order and Driver. Order attributes were

- Location of pick-up and drop-off
- Pick-up: urgent or booked in advance (for a certain date and time)
- Type of service (standard car, minivan, VIP, etc.)
- Importance of service (a number from 0 to 100 depending upon the client)
- Special requirements (pet, child chair, etc.)

Driver attributes were

- Type of vehicle
- Capability to complete special jobs
- Driver experience (novice or experienced)
- Domicile of driver
- Current vehicle location (GPS coordinates)
- Driver status (unavailable, break, working, free, will be free in 5/10 minutes, home transit)

Factual data on Object Instances (Individual Orders and Drivers/Vehicles statuses) were stored in client's databases, including Scenes (i.e., instantaneous models of the taxi service yielding every vehicle location and driver availability).

Virtual World

In the basic version of the scheduler, the allocation of taxis to customers was done by the negotiation between Order Agents, assigned to customers, and Driver Agents, assigned to taxi drivers. Order Agents were active: they compiled lists of available vehicles and initiated negotiations with Driver Agents. Driver Agents, in this first version of the system were designed to be only reactive: they only replied to requests from Order Agents and implemented the option selected by an Order Agent.

In the extended version, hereafter described, Order Agents and Driver Agents competed with each other or co-operated, depending on what was best for the whole enterprise. In this version, in addition to Order and Driver Agents, some new types of agents, namely, External Events Agents, Regional Loading Agents, and Orders Allocation Agents were also used.

Agents were designed to use flexible decision-making criteria instead of direct priorities, which is valuable when there is a need to deal with different categories of clients. For example, if a VIP order arrived and there was only one

driver that fully corresponded to the specified requirements and if that driver was already assigned to another job, the system would nevertheless allocate the VIP order to this driver and initiate re-scheduling of the previously agreed matches, if required.

The system first attempted to maximize company profit. Then, other criteria that are important for the business were considered, such as the service level and driver working conditions. For example, when choosing from two approximately equal options the system allocated the order to the driver who had not received orders for a longer time, thus ensuring relatively fair distribution of orders.

This virtual agent-based scheduling system was designed to work effectively with human dispatchers. In a situation where one dispatcher takes a new order and schedules a vehicle to come from north to south to pick up a client, and another dispatcher independently schedules another vehicle to go from south to north for another order, the virtual agents can spot this schedule anomaly and recommend dispatchers change their decisions to be more effective.

To enable improved performance, the taxi allocation system functioned in short cycles rather than as an immediate reaction to every event. Between the cycles the system collected the events and placed them in a queue. During each cycle, the events from the queue were processed, one by one, and appropriate agents, in turn, were given control by a designated human system dispatcher. Each event thus initiated a chain of negotiations among virtual agents. When all events were processed and the system dispatcher was satisfied that the best possible schedule was produced for that cycle, the schedule perturbation was implemented in the real world, and the system fell asleep (was idled) until a new event arrived causing the initiation of the next cycle.

To decrease the dimensions of the decision space, a pre-matching mechanism was used, which determined the suitability of Order-Driver matching. This mechanism cuts off unpromising options.

The Order-Driver pairs were evaluated before the final decision was made. An evaluation mark was given to each option and good options were remembered so that the evaluations did not need to be repeated later. The evaluation mark was determined using a multi-criteria model and calculated as a sum of all criteria values multiplied by their (variable) weights.

The following criteria were used for option evaluation: distance to the order, predicted delay of the pick-up, if any, preferences of the driver, driver experience, distance of the driver to overloaded area (to utilize drivers from outlying districts), service level conformity, importance and priority of the order, driver's place in a queue (if he is waiting at an airport), driver's home address (if he is looking for an order to or from home).

Scheduling workflow included the following steps:

1. New order arrives and joins the event queue
2. Possibility of order scheduling is checked
3. A software agent is assigned to the order
4. All drivers that can complete this order are included in pre-matching
5. Evaluation of all Order-Driver pairs is done according to agreed criteria
6. The Order Agent requests order completion costs from selected Driver Agents. This cost includes the cost of transferring the order from the previously allocated driver, if any
7. The Driver Agent receives the information on the reallocation costs by sending a request to its current Order Agent
8. If the revised decision is better than the previous one, it is applied
9. Step 6 continues for all candidate drivers, for whom the initial evaluation (without transfers) was better than the current evaluation
10. If no further changes occur during the cycle, the event processing is considered finished

In order to achieve the best possible solution, the system continued to search for improvements in previously agreed Order-Driver matches until the last moment when it had to issue the instruction to a driver to fulfil an order (commitment time). During this time interval the Driver was considered to be available for new allocations, but only

if the new allocation improved specified performance indicators.

When required, Driver Agents attempted “to come to an agreement” with each other about proposed re-allocation of orders. Occasionally, compensation was offered to the Driver Agent who lost a good client in order to improve overall value of the business, and Driver Agent satisfaction, in particular. Very often the re-scheduling of allocated resources caused a wave of negotiations aimed at the resolution of conflicts between new and old orders. The length of the re-scheduling chain was limited only by the time required for a taxi to reach a customer in a busy city such as London, which normally was sufficient for several changes of the schedule.

To summarise, the system built a schedule and perpetually reviewed it, attempting to improve key performance indicators as long as the time for essential re-scheduling was still available.

The Commitment Time was dynamically calculated for each order taking into account the priority and service type of the order and some other parameters. The introduction of the dynamic Commitment Time resulted in the increase of the fleet effectiveness by reducing the average task completion time per driver.

An option was introduced for the system to distribute the fleet according to the order-flow forecasts. Having information about the current order-flow and distribution of orders in the past, the expected order-flow was extrapolated, enabling the system to generate short-term (30 minutes) forecasts, which were normally reasonably correct. Based on the forecast, the system sent text messages to unoccupied drivers with recommendations to stay in, or move to the region where an increased order flow was expected. This feature enabled an improved distribution of the fleet, reducing response times and idle miles and increasing the number of pick-ups.

In cases when forecasts envisaged a probability of a VIP order arrival at a significant distance from the point where drivers were advised to congregate, the system would recommend that a proximate driver to move closer to the likely order point, offering him/her a guaranteed next order in exchange for compliance. This was an important feature because there were usually enough proximate drivers to complete available orders in areas that were not overloaded, and productivity of work in overloaded areas determined the actual fleet effectiveness. The system was also designed with an option to temporary amend criteria for the allocation of orders to drivers (for example, to extend the area where drivers are allowed to search for orders) to enable drivers to reach critical locations without being intercepted by less important orders from nearby locations.

The forecasting functionality was supported by an agent-based dynamic data mining system, which was, in fact, another complex adaptive system cooperating with the complex adaptive scheduler.

In later versions the system was designed to detect and identify drivers that cheat, i.e., by deliberately providing the scheduler with false information to gain personal advantages. Recorded cases include attempts to

- Reduce their ultimate waiting time by reporting that they were already waiting in an airport queue when, in reality, they may still have been tens of miles from the airport
- Get an earlier next order by indicating “free in 10 minutes” at or near the beginning of a long assignment
- Receive orders in their home direction by indicating “going home” several times during a day.

To reduce cheating Driver Agents were designed to monitor drivers’ schedules and ignore their messages, when judged inappropriate.

The final version of the complex adaptive taxi service scheduler negotiated only with agents that were affected by a disruptive event and then modified only affected parts of the schedule. This capability was a key feature that improved overall effectiveness.

Connecting Virtual and Real Worlds

The Virtual World, which is a model of reality and resides in the scheduler, is connected with the Real World of customers, dispatchers, and drivers, as follows.

As customers ring the call centre or visit the website to place, modify or cancel an order, dispatchers enter the pertinent information into the system. Drivers communicate with the system using GPS, mobile phones, or specialised handheld devices, conveying information on their location, direction of travel, availability, etc., and, in turn, receive instructions to pick up customers.

Lessons Learned

The system began its operation and maintenance phase in March 2008, only 6 months from the beginning of the project.

Results were extremely good: 98.5 % of all orders were allocated automatically without dispatcher's assistance; the number of lost orders was reduced by up to 2 %; the number of vehicles idle runs was reduced by 22.5 %. Each vehicle was able to complete two additional orders per week spending the same time and consuming the same amount of fuel, which increased the yield of each vehicle by 5 – 7 %.

Time required to repay investments was 2 months from the beginning of the operation and maintenance phase. During the first month of operation the fleet utilization effectiveness was increased by 5 – 7 %, which represents potential additional revenue of up to 5 million dollars per year. Such realized additional income has benefited both the company and the taxi drivers. According to available statistics, since 2008 driver wages have increased by 9 %, and there is a possibility for an overall fleet growth.

Delayed pick-ups were reduced by a factor of 3 which considerably improved customer service. Urgent order average response time (from booking until taxi pick-up arrival) decreased to 9 minutes which is the best time among all taxi services in London. For high priority orders the response time is 5 – 7 minutes or less. Response time reductions are especially noticeable in overloaded areas.

Implementation of "on the way home" orders, an improved allocation mechanism, when compared with a previous system, gives 3 – 4 thousand miles reduction in daily fleet run, greatly benefiting both drivers and the city's ecology.

Further developments targeting business effectiveness improvements may include an analysis of vehicle movements to determine actual vehicle velocities that could improve courier service by increasing the number of orders per courier.

References

Works Cited

Rzevski, G., Skobelev, P., "Managing Complexity" WIT Press, New Forest, Boston, 2014. ISBN 978-1-84564-936-4.

Glaschenko, A., Ivaschenko, A., Rzevski, G., Skobelev, P. "Multi-Agent Real Time Scheduling System for Taxi Companies". Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Decker, Sichman, Sierra, and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary. ISBN: 978-0-9817381-6-1, pp. 29-35.

Primary References

Rzevski, G., Skobelev, P., "Managing Complexity" WIT Press, New Forest, Boston, 2014. ISBN 978-1-84564-936-4.

Additional References

None.

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

Denver Airport Baggage Handling System

This example was developed as a SE example for the SEBoK. It describes systems engineering (SE) issues related to the development of the automated baggage handling system for the Denver International Airport (DIA) from 1990 to 1995. The computer controlled, electrical-mechanical system was part of a larger airport system.

Description

In February 1995, DIA was opened 16 months later than originally anticipated with a delay cost of $500 million (Calleam Consulting Ltd. 2008). A key schedule and cost problem—the integrated automated baggage handling system—was a unique feature of the airport. The baggage system was designed to distribute all baggage automatically between check-in and pick-up on arrival. The delivery mechanism consisted of 17 miles of track on which 4,000 individual, radio-controlled carts would circulate. The $238 million system consisted of over 100 computers networked together, 5,000 electric eyes, 400 radio receivers, and 56 bar-code scanners. The purpose of the system was to ensure the safe and timely arrival of every piece of baggage. Significant management, mechanical, and software problems plagued the automated baggage handling system. In August 2005, the automated system was abandoned and replaced with a manual one.

The automated baggage system was far more complex than previous systems. As planned, it would have been ten times larger than any other automated system, developed on an ambitious schedule, utilized novel technology, and required shorter-than-average baggage delivery times. As such, the system involved a very high level of SE risk. A fixed scope, schedule, and budget arrangement precluded extensive simulation or physical testing of the full design. System design began late as it did not begin until well after construction of the airport was underway. The change management system allowed acceptance of change requests that required significant redesigns to portions of work already completed. The design did not include a meaningful backup system; for a system that required very high mechanical and computer reliability, this increased failure risks. The system had an insufficient number of tugs and carts to cope with the volume of baggage expected and this, along with severely limited timing requirements, caused baggage carts to jam in the tracks and for them to misalign with the conveyor belts feeding the bags. This resulted in mutilated and lost bags (Neufville 1994; Gibbs 1994).

The baggage system problems could be associated with the non-use or misuse of a number of systems engineering (SE) concepts and practices: system architecture complexity, project scheduling, risk management, change management, system analysis and design, system reliability, systems integration, system verification and validation/testing, and insufficient management oversight.

Summary

The initial planning decisions, such as the decision to implement one airport-wide integrated system, the contractual commitments to scope, schedule, and cost, as well as the lack of adequate project management (PM) procedures and processes, led to a failed system. Attention to SE principles and practices might have avoided the system's failure.

References

Works Cited

Calleam Consulting Ltd. 2008. *Case Study – Denver International Airport Baggage Handling System – An illustration of ineffectual decision making*. Accessed on September 11, 2011. Available at http://calleam.com/WTPF/?page_id=2086.

Neufville, R. de. 1994. "The Baggage System at Denver: Prospects and Lessons." *Journal of Air Transport Management*. 1(4): 229-236.

Gibbs, W.W. 1994. "Software's Chronic Crisis." *Scientific American*. September 1994: p. 72-81.

Primary References

None.

Additional References

DOT. 1994. "New Denver Airport: Impact of the Delayed Baggage System." US Department of Transportation (DOT), Research Innovation Technology Administration. GAO/RCED-95-35BR. Available at http://ntl.bts.gov/DOCS/rc9535br.html

Donaldson, A.J.M. 2002. "A Case Narrative of the Project Problems with the Denver Airport Baggage Handling Systems (DABHS)," Software Forensics Center technical Report TR 2002-01. Middlesex University, School of Computer Sciences. Available at http://www.eis.mdx.ac.uk/research/SFC/Reports/TR2002-01.pdf

---

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Global Positioning System

The Global Positioning System (GPS) case study was developed by the United States Air Force Center for Systems Engineering (AF CSE) located at the Air Force Institute of Technology (AFIT). The GPS is a space-based radio-positioning system. A constellation of twenty-four satellites, including three spares, comprise the overall system which provides navigation and timing information to military and civilian users worldwide. GPS satellites, in one of six Earth orbits, circle the globe every twelve hours, emitting continuous navigation signals on two different L-band frequencies. The system consists of two other major segments: a world-wide satellite control network, and the GPS user equipment that can either be carried by a human user or integrated into host platforms such as ships, vehicles, or aircraft.

This case study discussion is based on the original source (O'Brien and Griffin 2007) which provides useful insights into what we might consider a "traditional" SE application. A second Global Positioning System Case Study II looks at the same case study from the perspectives of System of Systems (SoS) (glossary) engineering and Enterprise Systems Engineering (ESE) (glossary).

Domain Background

When looking at the Global Positioning System (GPS), it would be difficult to imagine another system that relie s so heavily upon such a wide range of domains, with the possible exception of the World Wide Web (WWW). Additionally, the various systems operating within these domains must all function together flawlessly to achieve success. It is evident from reading this case study that it directly relates to the following domains:

- aerospace;
- space;
- communications; and
- transportation.

This is also an example of systems of systems (SoS) and is considered an innovative technology.

The GPS case study includes a detailed discussion of the development of the GPS and its components, as well as other applicable areas. The reader of this study will gain an increased understanding of the effect that GPS has on military and commercial industries in the context of the systems engineering support required to achieve success.

Case Study Background

The United States Air Force Center for Systems Engineering (AF CSE), established in 2002 at the Air Force Institute of Technology (AFIT), was tasked to develop case studies focusing on the application of systems engineering principles within various aerospace programs. The GPS case study (O'Brien and Griffin 2007) was developed by AFIT in support of systems engineering graduate school instruction. The cases are structured using the Friedman-Sage framework (Friedman and Sage 2003; Friedman and Sage 2004, 84-96), which decomposes a case into contractor, government, and shared responsibilities in the following nine concept areas:

1. Requirements Definition and Management
2. Systems Architecture Development
3. System/Subsystem Design
4. Verification/Validation
5. Risk Management
6. Systems Integration and Interfaces
7. Life Cycle Support
8. Deployment and Post Deployment
9. System and Program Management

The Friedman-Sage framework (2004) is provided in Appendix A of the case study. This case study is an example where the government - specifically the JPO Systems Engineering Directorate - bore the responsibility for systems integration and configuration management. That is, the government played more than an oversight role in the systems engineering of the GPS system of systems. As mentioned in the case study, JPO developed the CONOPs, mission analysis, requirements and design analysis including security, and developed their own approach to the cryptology methodology. JPO coordinated the Configuration Control Board (CCB) chaired by the Program Director. JPO was also responsible for Level I ICDs and system design configurations; where the contractors were responsible for the system architecture and ICDs within their segment.

Case Study Description

The “Global Positioning System - Systems Engineering Case Study” describes the application of systems engineering during the concept validation, system design and development, and production phases of the GPS program (O'Brien and Griffin 2007). The case examines the applied systems engineering processes, as well as the interactions of the GPS joint program office (JPO), the prime contractors, and the plethora of government agencies that were associated with the program’s development and fielding. The systems engineering process is traced from the initiation of studies and the development of key technologies, which established the vision of a satellite navigation system in the 1960s, through to the multiphase joint-program that resulted in a fully operational capability release in 1995. This case study does not cover system enhancements incorporated through Blocks IIM, IIF, and III.

The GPS case study derived four learning principles (LPs) that explain the more broadly applicable areas of systems engineering knowledge that are addressed by the case study. These four LPs relate strongly to the SEBoK in the following areas:

- enabling individuals (LP1);
- configuration management (LP2);
- enabling the organization (LP3); and
- risk management (LP4).

Additionally, the GPS case study contains a thorough overview of life cycle management and exemplifies systems thinking principles.

Enabling Individuals

Learning Principle 1: Programs must strive to staff key positions with domain experts.

From the program management team, to the systems engineering, design, manufacturing, and operations teams, the individuals on the program were well-versed in their disciplines and all possessed a systems view of the program. While communications, working relationships, and organization were important, it was the ability of the whole team at all levels to understand the implications of their work on the system that was vital. Their knowledge-based approach for decision making had the effect of shortening the decision cycle because the information was understood and the base and alternative solutions were accurately presented.

Configuration Management

Learning Principle 2: The systems integrator must rigorously maintain program baselines.

The joint program office (JPO) retained the role of managing and controlling the system specification and, therefore, the functional baseline. The JPO derived and constructed a mutually agreed to set of system requirements that became the program baseline in 1973. While conducting the development program, the GPS team was able to make performance, risk, cost, and trade analyses against the functional baseline to control both risk and cost. The JPO was fully cognizant of the implications of the functional requirements on the allocated baseline because they managed the interface control working group process. Managing that process gave them first-hand knowledge and insight into the

risks at the lowest level. The individual with the system integrator role must rigorously maintain the system specification and functional baseline. There must be appropriate sharing of management and technical responsibilities between the prime contractor and their government counterparts to ensure success.

Enabling the Organization

Learning Principle 3: Achieving consistent and continuous high-level support and advocacy helps funding stability, which impacts systems engineering stability.

Consistent, continuous high-level support provides the requirements and assists funding stability. In this role, the Office of the Secretary of Defense (OSD) provided advocacy and sourced the funding at critical times in the program, promoted coordination among the various services, and reviewed and approved the GPS JPO system requirements. The OSD played the central role in the establishment and survivability of the program. The GPS JPO had clear support from the Director of Defense Development, Research, and Engineering, Dr. Malcolm Currie, and program support from the Deputy Secretary of Defense, Dr. David Packard. Clearly, the armed services – particularly the Navy and the Air Force early on, and later the Army – were the primary users of GPS and the eventual customers. However, each armed service had initial needs for their individual programs, or for the then-current operational navigation systems. Additionally, the secretary of the Air Force provided programmatic support to supply manpower and facilities.

Risk Management

Learning Principle 4: Disciplined and appropriate risk management must be applied throughout the life cycle.

The GPS program was structured to address risk in several different ways throughout the multiphase program. Where key risks were known up front, the contractor and/or the government utilized a classic risk management approach to identify and analyze risk, as well as develop and track mitigation actions. These design (or manufacturing/launch) risks were managed by the office who owned the risks. Identified technical risks were often tracked by technical performance measures (such as satellite weight and software lines of codes) and addressed at weekly chief engineer's meetings.

Serving in the clear role of program integrator allowed the JPO to sponsor risk trade studies at the top level. The JPO would issue study requests for proposals to several bidders for developing concepts and/or preliminary designs. Then, one contractor would be down-selected and the process would continue. This approach provided innovative solutions through competition, as well as helped in defining a lower risk, more clearly defined development program for the fixed-price contracts approach that was being used for development and production.

As the system integrator, the JPO was also closely involved with technical development. To identify unforeseeable unique technical challenges, the JPO would fund studies to determine the optimal approaches to new issues. There were schedule risks associated with the first launch due to unforeseen Block II issues with respect to the space vehicle and control segments (software development). Although a catastrophic event, the Challenger accident actually provided much needed schedule relief. Using decision analysis methodology led the JPO to an alternative approach to develop the expendable launch vehicle for the Block II satellites.

Good communication, facilitated by cooperative working relationships, was a significantly positive (though intangible) factor in the success of the GPS program, regardless of whether it was between the contractors and the government (JPO or other agencies), or between contractors and sub-contractors. A true team environment also played a significant role in reducing risk, especially considering the plethora of government agencies and contractors that were involved in the effort.

Life Cycle Management

The GPS case study takes the reader through the initial concept of GPS (March 1942) all the way to the development, production, and operational capability of the system. The current GPS program traces its heritage to the early 1960s when Air Force Systems Command initiated satellite-based navigation systems analyses conducted by The Aerospace Corporation. The case study follows the execution of the GPS program from the inception of the idea to the full operational capability release on April 27th, 1995. The concentration of the case study is not limited to any particular period, and the learning principles come from various times throughout the program's life.

Systems Thinking

The GPS case study highlights the need for systems thinking throughout. GPS satellites, in one of six Earth orbits, circle the globe every twelve hours. These satellites emit continuous navigation signals on two different L-band frequencies. The system consists of two other major segments: a world-wide satellite control network and the GPS user equipment that can either be carried by a human user, or integrated into host platforms such as ships, vehicles, or aircraft. The ability to conceive, develop, produce, field, and sustain the GPS demands the highest levels of systems thinking.

Summary

The GPS case study is useful for global systems engineering learning and provides a comprehensive perspective on the systems engineering life cycle. The study is applicable for detailed instruction in the following areas:

- enabling individuals;
- configuration management;
- enabling the organization;
- risk management;
- life cycle management; and
- systems thinking.

The GPS case study revealed that key Department of Defense personnel maintained a clear and consistent vision for this unprecedented, space-based navigation capability. The case study also revealed that good fortune was enjoyed by the JPO as somewhat independent, yet critical, space technologies matured in a timely manner.

Although the GPS program required a large degree of integration, both within the system and external to the system amongst a multitude of agencies and contractors, the necessary efforts were taken to achieve success.

Lastly, the reader of the GPS case study will gain an increased understanding of the effect that GPS has on the military and commercial industries in the context of the systems engineering support required to achieve success. The system was originally designed to help "drop five bombs in one hole" which defines the accuracy requirement in context-specific terms. The GPS signals needed to be consistent, repeatable, and accurate to a degree that, when used by munitions guidance systems, would result in the successful delivery of multiple, separately-guided munitions to virtually the identical location anywhere at any time across the planet. Forty to fifty years ago, very few outside of the military recognized the value of the proposed accuracy and most non-military uses of GPS were not recognized before 1990. GPS has increasingly grown in use and is now used every day.

References

Works Cited

Friedman, G.R. and A.P. Sage. 2003. Systems Engineering Concepts: Illustration Through Case Studies. January 19, 2003. Accessed September 2011. Available at: http://www.afit.edu/cse/docs/Friedman-Sage%20Framework.pdf.

Friedman, G. and A. Sage. 2004. "Case Studies of Systems Engineering and Management in Systems Acquisition." *Systems Engineering.* 7(1): p. 84-96.

O'Brien, Patrick J., and John M. Griffin. 4 October 2007. Global Positioning System. Systems Engineering Case Study. Air Force Center for Systems Engineering (AFIT/SY) Air Force Institute of Technology (AFIT). 2950 Hobson Way, Wright-Patterson AFB OH 45433-7765

Primary References

O'Brien, Patrick J., and John M. Griffin. 4 October 2007. Global Positioning System. Systems Engineering Case Study. Air Force Center for Systems Engineering (AFIT/SY) Air Force Institute of Technology (AFIT). 2950 Hobson Way, Wright-Patterson AFB OH 45433-7765

Additional References

none.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Global Positioning System II

This article highlights some of the differences between the so-called classical, traditional, or conventional systems engineering (SE) approaches and the newer, and, as yet, less defined principles of system of systems (SoS) engineering (SoSE). enterprise systems engineering (ESE), and/or complex systems engineering (CSE) or complex adaptive systems engineering (Gorod et al. 2015). The topic is still somewhat controversial, especially considering those that are sceptical that broader views of SE might work better when one is immersed in trying to cope with our most difficult problems. Indeed, the lack of a unified theory of SE is one of the prime motivations for producing and analysing case studies to develop more knowledge of what seems to work, what does not seem to work, and reasons why, really challenging SE environments.

For addition information, refer to Systems Engineering: Historic and Future Challenges, Systems Engineering and Other Disciplines, Enterprise Systems Engineering, and System of Systems Engineering.

Rather than modifying the previous discussion of the Global Positioning System Case Study in SEBoK, the focus is on comparing and contrasting the older and newer forms of SE by commenting on quotations from the original case study source documents (O'Brien and Griffin 2007).

Preface

The original case study begins by describing systems engineering (SE) principles. For example,

> System requirements are critical to all facets of successful system program development. First, system development must proceed from a well-developed set of requirements. Second, regardless of the evolutionary acquisition approach, the system requirements must flow down to all subsystems and lower-level components. And third, the system requirements must be stable, balanced, and must properly reflect all activities in all intended environments. However, system requirements are not unchangeable. As the system design proceeds, if a requirement or set of requirements is proving excessively expensive to satisfy, the process must rebalance schedule, costs, and performance by changing or modifying the requirements or set of requirements. (O'Brien and Griffin 2007, p. 9)

The Global Positioning System (GPS), including its multi-various applications, was developed over many years as the result of the efforts of a host of contributors. It is very difficult to believe that the classical, traditional or conventional systems engineering approach described in the above paragraph (especially those phrases highlighted in bold by the present authors) was truly responsible for this remarkable achievement that so profoundly impacts our lives. Rather, some more advanced form of systems engineering (SE), that might be called, system of systems (SoS) engineering (SoSE), enterprise systems engineering (ESE), or complex (adaptive) systems engineering (CSE), or a blend and/or combination of these approaches or methodologies, had to be responsible. This premise is supported explicitly and repeatedly in the following case study revision using bold font.

Continuing, the following quoted paragraphs seem flawed in several places highlighted in bold. The bold phrases might be replaced by the phrases in brackets [...]. Such brackets might also include other editorial comments of the present authors.

> Systems engineering includes making key system and design trades early in the process to establish the system architecture. These architectural artifacts This architecture can depict any new system, legacy system, modifications thereto, introduction of new technologies, and overall system-level behavior and performance. Modeling and simulation are generally employed to organize and assess architectural system alternatives at this stage. System and subsystem design follows the functional [system] architecture [as defined from a functional point of view]. System architectures designs are modified if elements are too risky, expensive, or time-consuming. (O'Brien and Griffin 2007, p. 9)

A good architecture, once established, should guide systems development, and not change very much, if at all, at least compared to possible changes in the system design, which, of course, can evolve as one learns more about the problem and potential solutions that may increase the system's capability. Thus, it is crucial to not confuse architecture with designs instantiating the architecture, contrary to what seems to be the case in (Ricci, et al. 2013).

> Important to the efficient decomposition and creation of functional and physical architectural designs are the management of interfaces and the integration of subsystems. interface management and integration is applied to subsystems within a system or across a large, complex system of systems. Once a solution is planned, analyzed, designed, and constructed, validation and verification take place to ensure satisfaction of requirements. Definition of test criteria, measures of effectiveness (MOEs), and measures of performance (MOPs) are established as part of the requirements process, taking place well before any component/subsystem assembly design and construction occurs. (O'Brien and Griffin 2007, p. 10)

In the quoted paragraph just above bold phrases note the emphasis on a reductionist approach, reductionism, where great attention is paid to the subsystems and managing the interfaces among them. This is the antithesis of a holistic approach where one concentrates on the whole system, recognizing that it is difficult to identify overall system behavior as depending on any particular subsystem or set of subsystems. In a truly complex system that is continually evolving, the above-mentioned requirements process is flawed because the system is continually changing, i.e., the system is evolutionary; the requirements are either ill-defined at the outset, or are modified

because stakeholders change their minds, or become somewhat irrelevant because the system environment changes.

> There are several excellent representations of the [usual traditional or conventional] systems engineering process presented in the literature. These depictions present the current state of the art in maturity and evaluation of the systems engineering process. One can find systems engineering process definitions, guides, and handbooks from the International Council on Systems Engineering (INCOSE), European Industrial Association (EIA), Institute of Electrical and Electronics Engineers (IEEE), and various Department of Defense (DoD) agencies and organizations. They show the process as it should be applied [Really? In all situations?] by today's experienced practitioner. One of these processes, long used by the Defense Acquisition University (DAU), is [a model] not accomplished in a single pass. This iterative and nested process gets repeated to the lowest level of definition of the design and its interfaces. (O'Brien and Griffin 2007, p. 10)

The above description appears to be written with pride without any acknowledgement that this SE methodology might fail to work if applied according to these guidelines, or that there might be new SE techniques that could be more effective in some situations. Again, this reflects a reductionist approach that ignores holism and emergent properties that might not be explained even when thoroughly understanding the systems components and their interactions. On the positive side, the next paragraph suggest how the world is changing and hints that something more is needed. Nevertheless, the advice seems to be oriented toward applying the existing SE discipline more vigorously instead of seeking new methods that might be more effective.

> The DAU model, like all others, has been documented in the last two decades, and has expanded and developed to reflect a changing environment. Systems are becoming increasingly complex internally and more interconnected externally. The process used to develop aircraft and systems of the past was effective at the time. It served the needs of the practitioners and resulted in many successful systems in our inventory. Notwithstanding, the cost and schedule performance of the past programs are replete with examples of well-managed programs and ones with less-stellar execution. As the nation entered the 1980s and 1990s, large DoD and commercial acquisitions experienced overrunning costs and slipping schedules. The aerospace industry and its organizations were becoming larger and were more geographically and culturally distributed. Large aerospace companies have worked diligently to establish common systems engineering practices across their enterprises. However, because of the mega-trend of teaming in large (and some small) programs, these common practices must be understood and used beyond the enterprise and to multiple corporations. It is essential that the systems engineering process govern integration, balance, allocation, and verification, and be useful to the entire program team down to the design and interface level. (O'Brien and Griffin 2007, p. 11)

Finally, in the next paragraph there is a suggestion that SE could be made more sophisticated but there is no mention of addressing people problems or advocating a broader transdisciplinary approach.

> Today, many factors overshadow new acquisition; including system-of-systems (SoS) con- text, network centric warfare and operations, and rapid growth in information technology. These factors are driving a more sophisticated systems engineering process with more complex and capable features, along with new tools and procedures. One area of increased focus of the systems engineering process is the informational systems architectural definitions used during system analysis. This process, described in DoD Architectural Framework (DoDAF), emphasizes greater reliance on reusable architectural views describing the system context and concept of operations, interoperability, information and data flows, and network service-oriented characteristics. (O'Brien and Griffin 2007, p. 11)

The last two sections of the systems engineering principles portion of the original case study address case studies themselves, mainly for academic purposes, to help people appreciate systems engineering principles, and the framework used in the case study, namely the rather narrowly defined Friedman-Sage framework that will be discussed briefly in Section II below.

The treatment of the reason for case studies is quite good in that it talks about the benefits of applying systems engineering principles, as highlighted from real-world examples of what works and what does not. Except near the end, where there is allusion to the possibility of new endeavor systems engineering principles, the principles espoused tend to be traditional or conventional.

On the other hand, based upon the original case study (O'Brien and Griffin 2007), if one views the boundary of the GPS system to include primarily the technology associated with the GPS space segment and its controlling ground network, then it can be assumed that system was likely implemented primarily by following traditional or conventional systems engineering processes. If one takes this viewpoint, then all of the above criticism which attempts to point out some of the shortcomings of conventional systems engineering, may seem vacuous at best, or politically incorrect at worst. It may well be that many would rather not denigrate the original GPS case study by exposing it to the possibilities of a broader system engineering approach.

Unless otherwise indicated, as the present authors have already been doing, unchanged quotations from the existing SEBoK are indented below. Modifications to such quotations are shown in brackets [...]; deletions are not necessarily shown explicitly.

Background

> The Global Positioning System (GPS) case study was developed by the United States Air Force Center for Systems Engineering (AF CSE) located at the Air Force Institute of Technology (AFIT). The GPS is a space-based radio-positioning system. A constellation of twenty-four satellites, including three spares, comprise the overall system which provides navigation and timing information to military and civilian users worldwide. GPS satellites, in one of six Earth orbits, circle the globe every twelve hours, emitting continuous navigation signals on two different L-band frequencies. The system consists of two other major segments: a world-wide satellite control network, and the GPS user equipment that can either be carried by a human user or integrated into host platforms such as ships, vehicles, or aircraft.

A user needs to receive signals from at least four GPS satellites simultaneously (satellite orbital positions and terrestrial terrain blockage can be issues that degrade performance) to determine one's position in three dimensions; the altitude determination is typically less accurate than the other two dimensions.

> When looking at [GPS], it would be difficult to imagine another system that relies so heavily upon such a wide range of [domains containing systems that must interact effectively to achieve successful GPS operation]. It is evident that [GPS directly relates to many domains and applications including:

- position location and tracking
- time synchronization
- navigation
- transportation
- times of arrival
- air traffic management
- situational awareness
- jam-resistant communications
- business and commerce
- farming
- aerospace
- sensing nuclear detonations from space
- military war-fighting
- targeting
- weapons delivery
- etc.].

> [GPS is] an example of [a collaborative (Dahmann, et al. 2008) systems of systems (SoS)]. As such, no one is in charge, and the capabilities (not requirements) flow from the bottom-up, as opposed to top-down.

Purpose

> The GPS case study includes a detailed discussion of the development of the GPS and its components, as well as other applicable areas. The reader of this study will gain an increased understanding of the effect that GPS has on military and commercial industries in the context of the systems engineering support required to achieve success.

This may be, but the principal purpose of this revised case study is to suggest a broader view of GPS that discusses signature aspects of SoS, enterprises, and complex systems, and emphasizes SoSE, ESE, and CSE.

> [AF CSE] was tasked to develop case studies focusing on the application of [SE] principles within various aerospace programs. The GPS case study [was developed in support of SE] graduate school instruction using the Friedman-Sage framework (Friedman and Sage 2003) (Friedman and Sage 2004).]

However, the Friedman-Sage framework involves only two contractual stakeholders, the Government and the contractor; further, the framework is limited to the traditional or conventional SE life cycle which mainly treats activities in a linear instead of nonlinear fashion; still further, only risks are considered, not a balance of risk and opportunity. Thus, the present authors believe a broader framework embracing SoSE, ESE, and CSE is more appropriate.

Challenges

In the original case study the first highly technical section (Section 2) was the system description. The original idea derived from trying to determine the precise orbital parameters of the first artificial satellites such as Sputnik launched by the Soviets in 1957. Researchers at Johns Hopkins realized the inverse, that if one knew precisely the orbital parameters, the locations of ground stations receiving satellite signals could be determined quite accurately. (O'Brien and Griffin 2007, p. 20)

GPS got its start in the early 70s (O'Brien and Griffin 2007, p. 19) building upon several previous satellite navigation systems. The primary motive was very accurate position information for the purposes of military applications. For example, the U.S. Air Force wanted to deliver nuclear weapons from bombers with unprecedented accuracy and precision. (O'Brien and Griffin 2007, p. 29)

With such an intense interest from the military, the first real challenge, other than the many technical challenges of making GPS work as well as envisioned, might have been the question of how to make GPS available to the civilian community so they could share the benefits. The study claimed that the system was always offered for civilian use, albeit with some charge. After the Korean airliner went astray and got shot down by a Soviet interceptor aircraft, President Reagan made GPS officially available for civilian use free of charge. (O'Brien and Griffin 2007, p. 14)

The second challenge could be associated with preserving precision capabilities for the military only, and relegating course acquisition (C/A) accuracy to the civilian community. (O'Brien and Griffin 2007, p. 15) Later this dichotomy was essentially eliminated with the realization that a differential GPS configuration involving a fixed ground station with a precisely known location will yield great accuracy. (Kee, et al. 1991)

The GPS satellites used space-borne atomic clocks. To alleviate the need for updating these clocks too often a successful effort was initiated to revise the international time standard which ended up using relatively infrequent "leap seconds". (O'Brien and Griffin 2007, p. 23) Even these are still annoying for many other applications, such as the continual need to achieve precise synchronization of frequency hopping radios.

An organizational challenge of inter-service rivalries was overcome with the formation of the Joint Program Office (JPO). (O'Brien and Griffin 2007, p. 25)

In the early days of satellite communication systems, for example, the satellites were quite small and low powered while the terminals were large and high-powered. By the time GPS came along, the satellites are getting bigger and more sophisticated. Then the challenge to develop relatively low-cost terminals, particularly for mobile users, greatly increased. (O'Brien and Griffin 2007, p. 29)

A small but interesting challenge was the definition of system of systems (SoS). It was decided that GPS was an SoS because it involved three independent systems, namely, the space vehicle (SV), the control segment (CS), and the user equipment (UE), that "merely" had to interface with each other. (O'Brien and Griffin 2007, p. 30)

Continually changing requirements is usually a problem, although in this case the requirements did not change as often as they could have. (O'Brien and Griffin 2007, p. 31)

Difficulties of defining and updating the many GPS interfaces was largely overcome by the GPS program director, Col. Brad Parkinson, when he convinced his own management, Gen. Schultz at Space and Missile Systems Office (SAMSO) (which eventually became the Space Division) that GPS ought to be defined solely by the signal-structure-in-space and not the physical interfaces. (O'Brien and Griffin 2007, p. 31)

Systems Engineering Practices

> Although the systems engineering process in Phase I has been discussed previously, this section will expand on the concepts. For example, one of the user equipment contractors was technically competent, but lacked effective management. The JPO strongly suggested that a systems engineering firm be hired to assist the contractor in managing program and they agreed. (O'Brien and Griffin 2007, p. 42)

There did not seem to be any mention of what SE firm was hired, if any. The Aerospace Corporation, a non-profit Federally Funded Research and Development Center (FFRDC), which had such a key role in the run-up to GPS was also prominently and centrally involved in development phase of this humungous project. (O'Brien and Griffin 2007, pp. 20, 22, 25, 33, 34, 40, 41, 44, 48, 50-52, 56, 57, 62, 63, 64, 66, 67, 71)

Lessons Learned

> Communications was a key ingredient that was fostered throughout GPS development. (O'Brien and Griffin 2007, p. 71)

Yes, from reading the original case study there seems to have been a lot of cooperation among the various organizations, more so than might have been expected in a less compelling case.

> Several precepts or foundations of the Global Positioning Satellite program are the reasons for its success. These foundations are instructional for today's programs because they are thought-provoking to those who always seek insight into the program's progress under scrutiny. These foundations of past programs are, of course, not a complete set of necessary and sufficient conditions. For the practitioner, the successful application of different systems engineering processes is required throughout the continuum of a program, from the concept idea to the usage and eventual disposal of the system. Experienced people applying sound systems engineering principles, practices, processes, and tools are necessary every step of the way. Mr. Conley, formerly of the GPS JPO, provided these words: "Systems engineering is hard work. It requires knowledgeable people who have a vision of the program combined with an eye for detail." (O'Brien and Griffin 2007, p. 72)

In very complex systems engineering efforts of this type, it is also important to explore new techniques that attempt to deal with "soft" issues involving people. Those that seem to work can be added to the systems engineering process collection.

> Systems engineering played a major role in the success of this program. The challenges of integrating new technologies, identifying system requirements, incorporating a system of systems approach, interfacing with a plethora of government and industry agencies, and dealing with the lack of an

operational user early in the program formation required a strong, efficient systems engineering process. The GPS program embedded systems engineering in their knowledge-base, vision, and day-to-day practice to ensure proper identification of system requirements. It also ensured the allocation of those requirements to the almost-autonomous segment developments and beyond to the subcontractor/vendor level, the assessments of new requirements, innovative test methods to verify design performance to the requirements, a solid concept of operations/mission analysis, a cost-benefit analysis to defend the need for the program, and a strong system integration process to identify and control the “hydra” of interfaces that the program encountered. The program was able to avoid major risks by their acquisition strategy, the use of trade studies, early testing of concept designs, a detailed knowledge of the subject matter, and the vision of the program on both the government and contractor side. (O’Brien and Griffin 2007, p. 72)

This well summarizes the successful systems engineering approach utilized in GPS. Another element of achieving overall balance is the pursuit of opportunities as the “flipside” of risk mitigation.

Finally, here is the list of academic questions offered in original case study.

QUESTIONS FOR THE STUDENT (O’Brien and Griffin 2007, p. 73) The following questions are meant to challenge the reader and prepare for a case discussion.

- Is this program start typical of an ARPA/ DARPA funded effort? Why or why not?
- Have you experiences similar or wildly different aspects of a Joint Program?
- What were some characteristics that should be modeled from the JPO?
- Think about the staffing for the GPS JPO. How can this be described? Should it be duplicated in today’s programs? Can it?
- Was there anything extraordinary about the support for this program?
- What risks were present throughout the GPS program. How were these handled?
- Requirement management and stability is often cited as a central problem in DoD acquisition. How was this program like, or [un]like, most others?
- Could the commercial aspects of the User Equipment be predicted or planned? Should the COTS aspect be a strategy in other DoD programs, where appropriate? Why or why not?

Other questions might be: What possible influences did the demand for or offering to the public of this GPS capability entail?What differences in the development of GPS might have emerged if the public was more aware of the potential applications for their benefit at the outset?

References

Works Cited

Dahmann, J. S., George Rebovich, Jr., and Jo Ann Lane. November 2008. “Systems Engineering for Capabilities.” CrossTalk, The Journal of Defense Software Engineering. http://www.stsc.hill.af.mil/crosstalk/2008/11/index.html.Accessed 12 May 2015).

Friedman, G.R., and A.P. Sage. 19 January 2003. Systems Engineering Concepts: Illustration Through Case Studies. Accessed September 2011. http://www.afit.edu/cse/docs/Friedman-Sage%20Framework.pdf.

Gorod, A., B. E. White, V. Ireland, S. J. Gandhi, and B. J. Sauser. 2015. Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. Boca Raton, FL: CRC Press, Taylor & Francis Group. 2015. http://www.taylorandfrancis.com/books/details/9781466502390/.Accessed 8 May 2015

Friedman, G.R., and A.P. Sage. 2004. “Case Studies of Systems Engineering and Management in Systems Acquisition.” Systems Engineering 7(1): 84-96.

Kee, Changdon, Bradford W. Parkinson, Penina Axelrad. Summer 1991. “Wide Area Differential GPS.” Journal of The Institute of Navigation 38(2): 123-46.

O’Brien, Patrick J., and John M. Griffin. 4 October 2007. Global Positioning System. Systems Engineering Case Study. Air Force Center for Systems Engineering (AFIT/SY) Air Force Institute of Technology (AFIT). 2950 Hobson Way, Wright-Patterson AFB OH 45433-7765.

Ricci, Nicola, Adam M. Ross, Donna H. Rhodes, and Matthew E. Fitzgerald. 2013. “Considering Alternative Strategies for Value Sustainment in Systems-of-Systems.” 6th IEEE International Systems Conference (SysCon). 15-18 April. Orlando, FL.

Primary References

Gorod, A., B. E. White, V. Ireland, S. J. Gandhi, and B. J. Sauser. 2015. Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. Boca Raton, FL: CRC Press, Taylor & Francis Group. 2015. http://www.taylorandfrancis.com/books/details/9781466502390/.Accessed 12 May 2015

Additional References

None

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

Next Generation Medical Infusion Pump

This case study summarizes the systems engineering aspects of the next-generation Symbiq™ IV (intravenous) medical pump development. Symbiq™ was developed by Hospira Inc. and documented in detail in Chapter 5 of the National Research Council book, *Human-System Integration in the System Development Process.* As described in the book, Symbiq™'s purpose was “to deliver liquid medications, nutrients, blood and other solutions at programmed flow rates, volumes and time intervals via intravenous and other routes to a patient, primarily for hospital use with secondary limited feature use by patients at home" (Pew 2007).

Domain Background

This case study provides insight into the use of systems engineering practices in a medical application.

Case Study Background

The project that is the subject of this report was approved by the Governing Board of the National Research Council, whose members are drawn from the councils of the National Academy of Sciences, the National Academy of Engineering, and the Institute of Medicine.

The study was supported by Award Nos. W911NF-05-0150 and FA5650-06-1-6610 between the National Academy of Sciences, the U.S. Department of the Army, and the U.S. Department of the Air Force.

Case Study Description

In creating a next-generation product, Hospira proposed to introduce new IV pump features, such as:

- multi-channel vs. single-channel liquid delivery;
- the ability to group multi-channeled devices together;
- associated user-programming capabilities and programmable drug libraries for specifying parallel delivery of liquids;
- use of color touchscreen devices;
- integration with numerous types of hospital information systems;
- ease of use for both medical personnel and patients at home;
- handling of potential hardware, software, and human-user faults;
- compliance with U.S. and international safety standards;
- use of alternating-current or battery power; and
- the ability to be cost-competitive and attractive to traditional medical and hospital administration personnel.

Many of these features are highly coupled, such as the multi-channel hardware controls, concurrent software synchronization, distinctive displays and alarms for multi-channel devices, and rigorous medical safety standards.

Views of the resulting medical infusion pump can be found as Figures 5-5 and 5-6 in Chapter 5, page 107 of the Pew and Mavor (2007) book. Systems engineering for the device involved a great deal of concurrent analysis and engineering of its hardware, software, human factors, operational, business, and safety aspects. It has been a commercial success and won the 2006 Human Factors and Ergonomics Society's User-Centered Product Design Award and the 2007 Medical Design Excellence Award.

Not only were there numerous technical challenges in the development of Symbiq™, but there were also challenges in the systems engineering of a product with a life-cycle operational concept that would produce satisfactory outcomes for a wide variety of product and operational stakeholders whose value propositions were often in some conflict. Some stakeholders wanted numerous features that would require a complex user interface, while others wanted a simple and easy to learn interface. Some users wanted the most advanced color touchscreen displays available, while others wanted a simpler, less-expensive product that was harder to misuse due to inadvertent screen commands. Some organizations felt that a minimal interpretation of the required safety features would be acceptable, while others advocated ultrahigh assurance levels. Some marketing personnel wanted a quick development and fielding of the basic product to capture market share, while maintainers wanted initial built-in life cycle support, maintenance, and diagnostic capabilities.

In such situations, many organizations focus on making quick requirement decisions and rapidly proceed into development. However, Hospira's understanding of the uncertainties and risks caused them to pursue a risk-driven, incremental commitment course of buying information to reduce risk, as emphasized in the SEBoK Part 3 knowledge area on Risk Management. As described in Pew and Mavor (2007), Hospira used a version of the Incremental Commitment Spiral Model (ICSM) summarized in the SEBoK Part 3 Knowledge Area on representative systems engineering process models. The following sections describe the project's incremental system definition progress through the ICSM exploration, valuation, foundations, and Development phases. Some evolution of terminology has occurred, the Pew and Mavor (2007) version uses ICM instead of ICSM and "architecting phase" instead of "foundations phase".

Symbiq™ Exploration Phase Summary

In the exploration phase, the project carried out numerous analyses on stakeholder needs, technical opportunities, and business competition. Using these analyses, the project team determined ranges of preferred options. Stakeholder needs analyses included contextual inquiry via shadowing of nurses using IV pumps and followup interviews, as well as creating task flow diagrams, use environment analyses, and user profiles analyses. Technical opportunity analyses included initial conceptual designs of multi-channel pump configurations, evaluation of commercially available single-color and multicolor display devices with touchscreen capabilities, and software approaches for specifying multi-channel delivery options and synchronizing concurrent processes.

Business competition analyses included hiring a management and marketing planning firm to examin next-generation pump competitor strengths and weaknesses with respect to such capabilities as the number of pump channels, therapies, programming options, air-in-line management, battery and alternating current capabilities, biomedical domain expertise, and alarms. Several key competitive advantages of a next-generation pump were identified, such as the ability to read bar-codes, small size, light weight, stand-alone functional channels, an extensive drug library, a high level of reliability, and clear mapping of screen displays and pumping channels.

Market research and market segment analyses also identified market windows, pricing alternatives, hospital purchasing decision-making trends, and safety aspects. These were iterated by focus groups of key thought leaders in critical care. The results were factored into a product concept plan, cost analysis, and business case analysis. These were independently reviewed by experts as part of the ICSM Valuation Phase Commitment Review process, which resulted in a go-ahead decision with an identification of several risks to be managed.

Symbiq™ Valuation Phase Summary

The valuation phase focused on the major risks highlighted in the Valuation Commitment Review, such as the multi-channel pump options, the types of programmable therapies, the need for tailorable medication libraries, the display screen and user interface options, and the safety considerations. The valuation phase also elaborated the product concept plan for the most attractive general set of options, including a development plan and operations plan, along with an associated cost analysis, risk analysis, and business case for review at the Foundations Commitment Review.

The multi-channel pump options were explored via several hardware industrial design mockups and early usability tests of the mockups. These included evaluation of such desired capabilities as semi-automatic cassette loading, special pole-mounting hardware, stacking of and total number of channels, and tubing management features. The evaluations led to the overall all choice to use a semi-automatic cassette loading capability with a red-yellow-green LED display to indicate concerns with the loading mechanism and with the pump in general.

Field exercises with prototypes of the pole mountings indicated the need for quick release and activation mechanisms, which were subsequently implemented. Risk analyses of alternative stacking mechanisms and the potential number of channels available established a preference for side-by-side stacking, a decision to develop one and two channel units, and to support a maximum of four channels in a stacked configuration.

The types of programmable therapies considered included continuous delivery for a specified time period, patient weight-based dosing, piggyback or alternating delivery between the two channels, tapered or ramped-rate delivery, intermittent-interval delivery, variable-time delivery, and multistep delivery. These were evaluated via prototyping of the software on a simulated version of the pump complexes and were iterated until satisfactory versions were found.

Evaluation of the tailorable medication libraries addressed the issue that different hard and soft safety limits were needed for dosages in different care settings (e.g., emergency room, intensive care, oncology, pediatric care, etc.) which creates a need for hospitals to program their own soft limits (overridable by nurses with permission codes) and hard limits (no overrides permitted). Stakeholder satisfaction with the tailoring features was achieved via prototype

exercises and iteration with representative hospital personnel.

A literature review was conducted to determine the relative advantages and disadvantages of leading input and display technologies, including cost and reliability data. After down-selecting to three leading vendors of touch screen color LCD displays and further investigating their costs and capabilities, a business risk analysis focused on the trade offs between larger displays and customer interest in small-footprint IV pumps. The larger display was selected based on better readability features and the reduced risk of accidental user entries since the larger screen buttons would help to avoid these occurrences. Extensive usability prototyping was done with hardware mockups and embedded software that delivered simulated animated graphic user interface (GUI) displays to a touchscreen interface that was integrated into the hardware case.

The safety risk analysis in the valuation phase followed ISO 14971:2000 standards for medical device design, focusing on Failure Modes and Effects Analyses (FMEAs). This analysis was based on the early high-level design, such as entry of excessive drug doses or misuse of soft safety limit overrides. Subsequent-phase FMEAs would elaborate this analysis, based on the more detailed designs and implementations.

As in the exploration phase, the results of the valuation phase analyses, plans, budgets for the succeeding phases, the resulting revised business case, evidence of solution feasibility, and remaining risks with their risk management plans were reviewed by independent experts and the ICSM Foundations Commitment Review was passed, subject to a few risk level and risk management adjustments.

Symbiq™ Foundations Phase Summary

During the foundations phase, considerable effort was focused on addressing the identified risks such as the need for prototyping the full range of GUI usage by the full range of targeted users, including doctors, home patients, the need for interoperability of the Symbiq™ software with the wide variety of available hospital information systems, and the need for fully detailed FMEAs and other safety analyses. Comparable added effort went into detailed planning for development, production, operations, and support, providing more accurate inputs for business case analyses.

GUI prototyping was done with a set of usability objectives, such as

- 90% of experienced nurses will be able to insert the cassette the first time while receiving minimal training;
- 99% will be able to correct any insertion errors;
- 90% of first time users with no training will be able to power the pump off when directed; and
- 80% of patient users would rate the overall ease of use of the IV pump three or higher on a five-point scale (with five being the easiest to use).

Similar extensive evaluations were done on the efficacy and acceptability of the audio alarms, including the use of a patient and intensive care unit simulator that included other medical devices that produced noises, as well as other distractions such as ringing telephones. These evaluations were used to enable adjustment of the alarms and to make the visual displays easier to understand.

Software interoperability risk management involved extensive testing of representative interaction scenarios between the Symbiq™ software and a representative set of hospital information systems. These resulted in several adjustments to the software interoperability architecture. Also, as the product was being developed as a platform for the next generation of infusion pump products, the software design was analyzed for overspecialization to the initial product, resulting in several revisions. Similar analyses and revisions were performed for the hardware design.

As the design was refined into complete build-to specifications for the hardware and the operational software, the safety analyses were elaborated into complete FMEAs of the detailed designs. These picked up several potential safety issues, particularly involving the off-nominal usage scenarios, but overall confirmed a high assurance level for the safety of the product design. However, the safety risk assessment recommended a risk management plan for the development phase to include continued FMEAs, thorough off-nominal testing of the developing product’s hardware

and software, and extensive beta-testing of the product at representative hospitals prior to a full release.

This plan and the other development and operations phase plans, product feasibility evidence, and business case analysis updates were reviewed at a Development Commitment Review, which resulted in a commitment to proceed into the development phase.

Symbiq™ Development Phase Systems Engineering Summary

The development phase was primarily concerned with building and testing the hardware and software to the build-to specifications, but continued to have an active systems engineering function to support change management; operations, production, and support planning and preparation; and further safety assurance activities as recommended in the risk management plan for the phase.

For hospital beta-testing, thoroughly bench-tested and working beta versions of the IV pump were deployed in two hospital settings. The hospitals programmed drug libraries for at least two clinical care areas. The devices were used for about four weeks. Surveys and interviews were conducted with the users to capture their "real world" experiences with the pump. Data from the pump usage and interaction memory was also analyzed and compared to the original doctors' orders. The beta tests revealed a number of opportunities to make improvements, including revision of the more annoying alarm melodies and the data entry methods for entering units of medication delivery time in hours or minutes.

Usability testing was also conducted on one of the sets of abbreviated instructions called TIPS cards. These cards serve as reminders for how to complete the most critical tasks. Numerous suggestions for improvement in the TIPS cards themselves, as well as the user interface, came from this work, including how to reset the "Air-in-Line" alarm and how to check all on-screen help text for accuracy.

The above mentioned usability objectives were used as Acceptance Criteria (glossary) for the validation usability tests. These objectives were met. For example, the calculated task completion accuracy was 99.66% for all tasks for first time nurse users with minimal training. There were a few minor usability problems uncovered that were subsequently fixed without major changes to the GUI or effects on critical safety related tasks.

The risk analysis was iterated and revised as the product development matured. FMEAs were updated for safety critical risks associated with three product areas: the user interface, the mechanical and electrical subsystems, and the product manufacturing process. Some detailed implementation problems were found and fixed, but overall the risk of continuing into full-scale production, operations, and support was minimal. Systems engineering continued into the operations phase, primarily to address customer change requests and problem reports, and to participate in planning for a broader product line of IV pumps.

Overall, customer satisfaction, sales, and profits from the Symbiq™ IV pump have been strong and satisfaction levels from the management, financial, customer, end user, developer, maintainer, regulatory, and medical-community stakeholders have been quite high (Pew 2007).

Summary

In summary, the Symbiq™ Medical Infusion Pump Case Study provides an example of the use of the systems engineering practices discussed in the SEBoK. As appropriate for a next-generation, advanced technology product, it has a strong focus on risk management, but also illustrates the principles of synthesis, holism, dynamic behavior, adaptiveness, systems approach, progressive entropy reduction, and progressive stakeholder satisfying discussed in Part 2 of the SEBoK. It provides an example of an evolutionary and concurrent systems engineering process, such as the incremental commitment spiral process, and of other knowledge areas discussed in SEBoK Parts 3 and 4, such as system definition, system realization, system engineering management, and specialty engineering.

References

Works Cited

Pew, R. and A. Mavor. 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Medical Radiation

This case study presents system and software engineering issues relevant to the accidents associated with the Therac-25 medical linear accelerator that occurred between 1985 and 1988. The six accidents caused five deaths and serious injury to several patients. The accidents were system accidents that resulted from complex interactions between hardware components, controlling software, and operator functions.

Domain Background

Medical linear accelerators, devices used to treat cancer, accelerate electrons to create high energy beams that can destroy tumors. Shallow tissue is treated with the accelerated electrons. The electron beam is converted to X-ray photons to reach deeper tissues. Accidents occur when a patient is delivered an unsafe amount of radiation.

A radiation therapy machine is controlled by software that monitors the machine's status, accepts operator input about the radiation treatment to be performed, and initializes the machine to perform the treatment. The software turns the electron beam on in response to an operator command. The software turns the beam off whenever the treatment is complete, the operator requests the beam to shutdown, or when the hardware detects a machine malfunction. A radiation therapy machine is a reactive system in which the system's behavior is state dependent and the system's safety depends upon preventing entry into unsafe states. For example, the software controls the equipment that positions the patient and the beam. The positioning operations can take a minute or more to execute, thus it is unsafe to activate the electron beam while a positioning operation is in process.

In the early 1980s, Atomic Energy of Canada (AECL) developed the Therac-25, a dual-mode (X-rays or electrons) linear accelerator that can deliver photons at 25 megaelectron volts (MeV) or electrons at various energy levels. The Therac-25 superseded the Therac-20, the previous 20-MeV dual mode accelerator with a history of successful clinical use. The Therac-20 used a DEC PDP-11 (Digital Equipment Corporation Programmed Data Processor) minicomputer for computer control and featured protective circuits for monitoring the electron beam, as well as mechanical interlocks for policing the machine to ensure safe operation. AECL decided to increase the responsibilities of the Therac-25 software for maintaining safety and eliminated most of the hardware safety mechanisms and interlocks. The software, written in PDP-11 assembly language, was partially reused from earlier products in the Therac product line. Eleven Therac-25s were installed at the time of the first radiation accident in June 1985.

The use of radiation therapy machines has increased rapidly in the last 25 years. The number of medical radiation machines in the United States in 1985 was approximately 1000. By 2009 the number had increased to approximately 4450. Some of the types of system problems found in the Therac-25 may be present in the medical radiation devices currently in use. References to more recent accidents are included below.

Case Study Background

The Therac-25 accidents and their causes are well documented in materials from the U.S. and Canadian regulatory agencies (e.g., the U.S. Food and Drug Administration (FDA) and the Canadian Bureau of Radiation and Medical Devices) and in depositions associated with lawsuits brought against AECL. An article by Leveson and Turner (1993) provides the most comprehensive, publicly available description of the accident investigations, the causes of the accidents, and the lessons learned relevant to developing systems where computers control dangerous devices.

Case Study Description

The Therac-25 accidents are associated with the non-use or misuse of numerous system engineering practices, especially system verification and validation, risk management, and assessment and control. In addition, numerous software engineering good practices were not followed, including design reviews, adequate documentation, and comprehensive software unit and integration tests.

The possibility of radiation accidents increased when AECL made the systems engineering decision to increase the responsibilities of the Therac-25 software for maintaining safety and eliminated most of the hardware safety mechanisms and interlocks. In retrospect, the software was not worthy of such trust. In 1983 AECL performed a safety assessment on the Therac-25. The resulting fault tree did include computer failures, but only those associated with hardware; software failures were not considered in the analysis.

The software was developed by a single individual using PDP-11 assembly language. Little software documentation was produced during development. An AECL response to the FDA indicated the lack of software specifications and of a software test plan. Integrated system testing was employed almost exclusively. Leveson and Turner (1993) described the functions and design of the software and concluded that there were design errors in how concurrent processing was handled. Race conditions resulting from the implementation of multitasking also contributed to the accidents.

AECL technical management did not believe that there were any conditions under which the Therac-25 could cause radiation overdoses, and this belief was evident in the company's initial responses to accident reports. The first radiation overdose accident occurred in June 1985 at the Kennestone Regional Oncology Center in Marietta, Georgia, where the Therac-25 had been operating for about 6 months. The patient who suffered the radiation overdose filed suit against the hospital and AECL in October 1985. No AECL investigation of the incident occurred and FDA investigators later found that AECL had no mechanism in place to follow up potential reports of suspected accidents. Additionally, other Therac-25 users received no information that an accident had occurred.

Two more accidents occurred in 1985, including a radiation overdose at Yakima Valley Memorial Hospital in Yakima, Washington that resulted in an accident report to AECL. The AECL technical support supervisor responded to the hospital in early 1986: "After careful consideration, we are of the opinion that this damage could not have been produced by any malfunction of the Therac-25 or by any operator error… there have apparently been no other instances of similar damage to this or other patients."

In early 1986 there were two accidents at the East Texas Cancer Center in Tyler, Texas, both of which resulted in the death of the patient within a few months. On March 21, 1986 the first massive radiation overdose occurred, though the extent of the overdose was not realized at the time. The Therac-25 was shut down for testing the day after the accident. Two AECL engineers, one from the plant in Canada, spent a day running machine tests but could not reproduce the malfunction code observed by the operator at the time of the accident. The home office engineer

explained that it was not possible for the Therac-25 to overdose a patient. The hospital physicist, who supervised the use of the machine, asked AECL if there were any other reports of radiation overexposure. The AECL quality assurance manager told him that AECL knew of no accidents involving the Therac-25.

On April 11, 1986 the same technician received the same malfunction code when an overdose occurred. Three weeks later the patient died; an autopsy showed acute high-dose radiation injury to the right temporal lobe of the brain and to the brain stem. The hospital physicist was able to reproduce the steps the operator had performed and measured the high radiation dosage delivered. He determined that data-entry speed during editing of the treatment script was the key factor in producing the malfunction code and the overdose. Examination of the portion of the code responsible for the Tyler accidents showed major software design flaws. Levinson and Turner (1993) describe in detail how the race condition occurred in the absence of the hardware interlocks and caused the overdose. The first report of the Tyler accidents came to the FDA from the Texas Health Department. Shortly thereafter, AECL provided a medical device accident report to the FDA discussing the radiation overdoses in Tyler.

On May 2, 1986 the FDA declared the Therac-25 defective and required the notification of all customers. AECL was required to submit to the FDA a corrective action plan for correcting the causes of the radiation overdoses. After multiple iterations of a plan to satisfy the FDA, the final corrective action plan was accepted by the FDA in the summer of 1987. The action plan resulted in the distribution of software updates and hardware upgrades that reinstated most of the hardware interlocks that were part of the Therac-20 design.

AECL settled the Therac-25 lawsuits filed by patients that were injured and by the families of patients who died from the radiation overdoses. The total compensation has been estimated to be over $150 million.

Summary

Leveson and Turner (1993) describe the contributing factors to Therac-25 accidents: “We must approach the problems of accidents in complex systems from a systems-engineering point of view and consider all contributing factors." For the Therac-25 accidents, the contributing factors included

- management inadequacies and a lack of procedures for following through on all reported incidents;
- overconfidence in the software and the resulting removal of hardware interlocks (causing the software to be a single point of failure that could lead to an accident);
- less than acceptable software engineering practices; and
- unrealistic risk assessments along with over confidence in the results of those assessments.

Recent Medical Radiation Experience

Between 2009 and 2011, The New York Times published a series of articles by Walter Bogdanich on the use of medial radiation, entitled “Radiation Boom" (2011).

The following quotations are excerpts from that series:

> *Increasingly complex, computer-controlled devices are fundamentally changing medical radiation, delivering higher doses in less time with greater precision than ever before.” But patients often know little about the harm that can result when safety rules are violated and ever more powerful and technologically complex machines go awry. To better understand those risks, The New York Times examined thousands of pages of public and private records and interviewed physicians, medical physicists, researchers and government regulators. The Times found that while this new technology allows doctors to more accurately attack tumors and reduce certain mistakes, its complexity has created new avenues for error — through software flaws, faulty programming, poor safety procedures or inadequate staffing and training. . . .*
>
> *Linear accelerators and treatment planning are enormously more complex than 20 years ago,’ said Dr. Howard I. Amols, chief of clinical physics at Memorial Sloan-Kettering Cancer Center in New York. But*

hospitals, he said, are often too trusting of the new computer systems and software, relying on them as if they had been tested over time, when in fact they have not. . . .

Hospitals complain that manufacturers sometimes release new equipment with software that is poorly designed, contains glitches or lacks fail-safe features, records show. Northwest Medical Physics Equipment in Everett, Wash., had to release seven software patches to fix its image-guided radiation treatments, according to a December 2007 warning letter from the F.D.A. Hospitals reported that the company's flawed software caused several cancer patients to receive incorrect treatment, government records show.

References

Works Cited

Bogdanich, W. 2011. Articles in the "Radiation Boom" series. *New York Times.* June 2009-February 2011. Accessed November 28, 2012. Available: http://topics.nytimes.com/top/news/us/series/radiation_boom.

Leveson, N.G., and C.S. Turner. 1993. "An Investigation of the Therac-25 Accidents." *Computer*. 26 (7): 18-41.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Project Management for a Complex Adaptive Operating System

This article is based on the **Complex Adaptive Operating System: Creating Methods for Complex Project Management** case study (Findlay and Straus, 2015). The case study focuses on creating tools and methods that project managers can use in managing complex adaptive systems projects.

Background

The International Centre for Complex Project Management (ICCPM) is a non-profit organization that was created to address "the international community's ability to successfully deliver very complex projects and manage complexity across all industry and government sectors" (ICCPM, 2012).

In an ongoing effort to help member organizations successfully undertake major complex projects, ICCPM conducted a bi-annual series of international round-tables. The purpose of the round-tables was to better understand what contributes to the success of complex projects and to identify and develop new and improved tools and approaches. The round-tables were facilitated using a computer-assisted collaborated meeting process that leverages the features of complex adaptive systems—described below—to help people with diverse viewpoints and experience create new collective understanding and plans for action.

Complex major projects are known for being unsuccessful in on-time and on-budget completion. An (IBM, 2008) survey of 1,500 change managers found that only 40% of projects finished on time and on budget. Barriers to success were the inability to change attitudes or mindsets (58%), dysfunctional culture (40%) lack of senior management support (32%) and underestimating the complexity of a project (35%).

However, several new systemic approaches show considerable promise as a way to think about and manage projects. Six frameworks help inform these approaches: systems thinking, the features of complex adaptive systems (CASs), complexity theory, the Complexity Model of Change (Findlay and Straus, 2011); polarity thinking as a way of thinking about and leveraging wicked problems, cognitive complexity and adult development theory.

Systems thinking recognizes whole systems and the interdependencies of their parts. A system may be defined as "a set of things, organisms, and people that, as a result of their interconnection, produce their own patterns of behavior over time" (Meadows, 2008, p.2). A system cannot be understood by focusing on its parts alone (Wheatley, 1999). To successfully address and influence a system, such as a complex project, one must understand the whole and how its parts interact.

Some project managers continue to think of a project as a "machine" that operates according to linear or algorithm rules. This persistent and largely unhelpful meme is now being displaced by a new and more robust model, which regards projects as complex adaptive systems (CASs). Unlike strictly mechanical systems, CASs are self-organizing, learn from experience, are emergent, and from time to time undergo large scale phase transitions to new and higher states of the system. This view of large-scale projects—which are heavily influenced by human interaction—is proving to be much more accurate and allows project managers to leverage techniques for exerting influence in environments that have previously presented intractable problems .

Leaders of complex projects would also be wise to consider three fundamental theorems of complexity theory, which apply to CASs and which are critical to project success. These are a robust model of the system, requisite variety and adopting solutions which act at an appropriate scale.

- The robust model axiom considers that "no one can effectively influence a system until they have a thorough understanding of its scope and the connections and interdependencies" (Conant and Ashby, 1970).
- The law of requisite variety contents that "complexity can only be dealt with by equal or greater complexity" (Ashby, 1956, p. 2). In other words, in order to deal effectively with the diversity of problems presented by a

complex project, one must have a repertoire of responses which is (at least) as nuanced as the problems themselves (Requisite Variety, 2015).

- The scale condition requires that those who wish to exercise leverage over a system must recognize that “highly complex situations can best be addressed by greater degrees of freedom at the local scale so that innovation and adaptability are maximized” (Bar-Yam, 2004).

A third framework, the Complexity Model of Change, is a model of socio-technological change, comprising a series of growth and decay curves or waves, which helps project managers better understand how to influence systems and design new ones, so the roles, methods, rules of interaction or engagement, technologies and relationships between people are better aligned with each other and the desired outcome. The overlapping waves represent large-scale eras, for example, the Industrial Age and the Information Age, which have at their core a metaphor, for example the machine and the computer. The current wave, the Knowledge Age, is based on a network metaphor. The wave we are now entering is the Wisdom Age, which began in 2010. Its core metaphor is the complex adaptive system and the main thrust of this period is the wise application of knowledge.

Another area that project managers may now address differently is that of wicked problems or paradoxes: problems that are ill-defined and recurrent, and which, when attempts are made to solve them with single optimal solutions, create another problem. Polarity thinking regards wicked problems as sets of interdependent values or ideas—like centralizing for efficiency and decentralizing for adaptability—that persist together over time and need each other for the success of the system. If we pay attention to one pole at the expense of the other we achieve sub-optimal results. When we manage polarities as a system, we realize the benefits of both poles and achieve high performance over time with a minimum of vacillation and the need for correction.

Other disciplines that are critical to project success are understanding and making best use of new ways of thinking about issues and relating to others in more flexible and adaptive ways. Theories of cognitive complexity and adult development theory can contribute to how we think about this problem. For example, “triple-loop learning” (Gragert, 2013), helps us think about issues from a higher level of cognitive complexity. Instead of asking are we doing things right (single loop), we might ask are we doing the right things (double loop) or how do we decide what is the most effective paradigm to use to influence and create benefit for the system (triple)?

Purpose

The purpose of the ICCPM roundtables was to help project managers develop new and better ways to lead complex major projects, by bringing together people from both the buy-side and the supply-side to share their knowledge and experience and to grow a network of practitioners, professionals, researchers, and educators able to deliver leading edge complex project management solutions to client organizations and partners around the world (Findlay and Straus, 2015, p. 489).

Challenges

There are many challenges to be addressed in the complex project management environments. The three top contenders are 1) developing new ways of thinking, acting, and interacting; 2) developing more robust models of the system by getting everyone in the room—the project management team and their stakeholders; and 3) steering projects through multiple disruptive socio-technological shifts using the feature of complex adaptive systems.

```
“People, their organizations, and their projects need to be capable of
reorganizing into new forms, which are a better fit with the new
context” (Findlay and Straus, 2015, p. 494).
```

Systems Engineering Practices

One of the tools Findlay and Straus use to deal with all three challenges in the context of group interaction, such as the ICCPM round-tables, is the Zing complex adaptive meeting process. The process is used to guide conversation in the room, to capture, simultaneously display ideas and to help participants integrate and make meaning from the ideas. The tool was used for the round-tables to help people work together in new ways, develop new and better models of the system together and to design and pilot new and better decision and learning methods.

The technology "provide[s] a container for a suitably representative sample of the people in the system to meet and conceptualize a robust model of the system and develop strategies for how to leverage the system" (Findlay and Straus, 2015, p. 492).

A "talk-type-read-review" (Findlay and Straus, 2015, p. 492) etiquette was employed to organize the session, which, in complexity theory terms, is a simple rule of interaction. Rich, open-ended questions guide the conversations, the ideas are read out aloud and the common themes or stand out ideas are recorded by the facilitators.

The open-ended questions are asked one at a time to explore all possibilities and reduce complexity. Although, round-table participants often held opposing views at the beginning, of the session, through a processes of continual, iterative feedback, they ultimately arrived at similar or complementary conclusions by the end of the round-table.

The process "automates"—or helps participants engage in—ways of interacting that incorporate a higher level of cognitive complexity than the participants might engage in individually, thus facilitating a shift in the group to a higher level of system performance.

Lessons Learned

The role of leaders of complex projects is to help their organization systems successfully deliver on time and on budget amidst constant change. Their mandate is to deliver amazing new solutions while making few or no mistakes—a challenging goal even in far less complex environments. In order to be successful, project leaders (and their teams) need new systems structures—new tools and methods—that reliably get better results. They need to have a robust and fresh understanding of the systems over which they preside and how they might influence them to greatest effect.

No longer can the complex project leader go off into a corner and design a project and then try to sell it to the community and political leaders, for example. Leaders now need to involve the whole system in the design of a project from its inception through to completion. They need to deal with wicked problems not by looking for the one best solution, but by integrating and leveraging competing ideas. This requires a shift in perspective: from attempting to "control" a complex project system as one might control a mechanical device, to understanding projects as highly complex and interconnected "living" systems that evolve over time. While we do not have "control" over our systems in the classical sense, we can exert influence very effectively, provided that we constantly update our understanding of what is going on and learn new ways to act and interact that are more likely to achieve our desired outcomes.

To achieve this, leaders need to develop the capacity to "anticipate the skills, leadership and coordination roles, technologies, methods, and processes that will be required to successfully surf the waves of change…" (Findlay and Straus, 2015, p. 501).

The 2012 ICCPM round-table series discussion paper (ICCPM, 2012) uses the example of a system undergoing transformation of many levels to illustrate the difficulty that complex project leaders face:

```
"The issue has been characterized as learning to fly a plane, while the
 plane is already in the air, and being re-assembled into another kind
of transportation technology altogether. And, at the same time, the
current passengers are disembarking and another group is boarding that
demands a better quality of service or experience at lower cost than
```

```
ever before. (ICCPM 2015 p. 21)”
```

This case study illustrates the need, in times of accelerating change, of “a real-time, systems-wide approach to the development of the methods and tools for managing complex projects” (Findlay and Straus, 2015, p. 500) so leaders can deal successfully and creatively with uncertainty and ambiguity.

References

Works Cited

Ashby, R. 1956. *An introduction to cybernetics*. London: Chapman and Hall.

Bar Yam, Y. 2004. *Making things work: Solving complex problems in a complex world*. Cambridge, MA: Knowledge Press.

Conant, R., and Ashby, R. 1970. *Every good regulator of a system must be a model of that system*. International Journal of System Sciences. 1(2): 89-97.

Findlay, J., and Straus, A. 2011. *A shift from systems to complex adaptive systems thinking*. In O. Bodrova and N. Mallory (Eds.), Complex Project Management Task Force Report: Compendium of Working Papers. Canberra: International Centre for Complex Project Management: 24-26.

Findlay, J., and Straus, A. 2015. *Complex Adaptive Operating System: Creating Methods for Complex Project Management*. In Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. Gorod, A., White, B.E., Ireland, V., Gandhi, S.J., and Sauser, B. (Eds.). Boca Raton: CRC Press: 471-505.

Gragert, T. 2013. *Triple loop learning*. Thorston’s site. Retrieved 12 June, 2015 from http://www.thorsten.org/wiki/index.php?title=Triple_Loop_Learning.

IBM. 2008 Making Change Work. New York: IBM.

ICCPM. 2012. *Roundtable discussion paper: Complexity in a time of global financial change: Program delivery for the new economy*. Findlay, J., Straus, A., and Hatcher, C. (Eds). Canberra: International Centre for Complex Project Management.

Meadows, D. 2008. *Thinking in Systems: A primer*. White River Junction, Vermont: Chelsea Green Publishing.

Requisite Variety. 2015. *What is requisite variety?*. Retrieved 1 June, 2015 from http://requisitevariety.co.uk/what-is-requisite-variety.

Wheatley, M. 1999. *Leadership and the new science: Discovering order in a chaotic world*. San Francisco: Berrett-Koehler.

Primary References

Findlay, J., and Straus, A. 2015. *Complex Adaptive Operating System: Creating Methods for Complex Project Management*. In Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. Gorod, A., White, B.E., Ireland, V., Gandhi, S.J., and Sauser, B. (Eds.). Boca Raton: CRC Press: 471-505.

Additional References

None.

Russian Space Agency Project Management Systems

This article is based around a **Russia space agency Project Management Systems case study** (Rzevski and Skobelev, 2014). The case study focuses on the development of a Real-Time Complex Adaptive Project Management Systems capable of effectively managing multiple related projects collectively contributing to a defined Enterprise Value.

Background

The essence of every management system should be the same: the allocation of human, physical, financial and intellectual (knowledge) resources to demands (tasks) with the aim of increasing the specified Enterprise Value, where the Enterprise Value is a system of values such as profit, market share, business sustainability, quality of service to customers, quality of working conditions for employees, quality of life in the community, etc. The key difference between the management of a business and management of a project is that businesses are continuously evolving processes whilst projects have specified beginnings and ends.

Standard challenges for project management are:

- Stringent budgets and deadlines
- High competition for limited resource availability such as up-to-date domain knowledge, skills and advanced productivity tools
- Functional organisation, which inevitably impedes interdepartmental cooperation
- Bureaucratic management, which is more concerned with lines of command and reporting than with the full use of project member’s initiative and creativity and which negatively affects their motivation
- Rigid project planning, which leads to a rapid divergence between the project plan and reality

Large enterprises commonly operate several projects concurrently. What is best for an individual project it is not always best for the enterprise and therefore it is necessary to implement coordination of concurrently run projects with the objective of significantly increasing Enterprise Value.

There are two new key problems, which the 21st century brought to us.

The first is the rapidly increasing complexity of the Internet-based global market, which creates frequent unpredictable disruptive events. This requires real-time adaptive project management, for which there is at present no precedent.

The second is the replacement of capital with knowledge as the key business resource in the economy in which the wealth created by knowledge services is greater than wealth created by producers of goods. There is at present no management system capable of discovering, processing, storing and allocating knowledge to project tasks.

Purpose

The client for this case study was one of the key space technology organisations in Russia, their equivalent of NASA, which operates, at any time, many concurrent mission critical projects.

The purpose of the project management system was to enable the client to effectively manage several related projects (at least ten), collectively contributing to the specified Enterprise Value, with the following requirements.

- Each project may consist of up to 5,000 constituent tasks
- Project members may have different backgrounds and skills and may belong to diverse business cultures
- Project members must have an opportunity to contribute to decision making processes, which affects their domain of work (distributed decision making)
- Both project management and project members must have readily available and up-to-date information on, respectively, project and individual progress, productivity and achievements of goals
- The allocation of resources to tasks must consider 4 types of resources, namely, human, physical, financial and knowledge
- Availability of resources for and constituent tasks of each project may change with short notice and these changes must be rapidly incorporated into the system
- Projects will be subjected to frequent disruptive events such as non arrival of expected orders, arrival of unexpected orders, sudden and unforeseen emergence of external/internal competitors, cancellations, changes in task specifications, delays, failures, no-shows, etc.
- Rules and regulations governing projects are likely to change rather frequently and any change in rules and regulations must be incorporated immediately and easily into the relevant project management system
- Any discrepancy between project plans and reality in the field must be continuously monitored and rapidly detected and reported
- Projects may cooperate and/or compete for resources in order to increase specified Enterprise Value

A thorough analysis of the client's requirements led to the conclusion that it is necessary to develop a real-time, complex adaptive project management system capable of cooperating and/or competing with other systems, with the overarching goal to continuously increase specified Enterprise Value.

The new system would replace a number of stand-alone, manual or semi-automated project management systems with inadequate monitoring of progress and productivity.

A thorough analysis of contemporary practices showed that such a transformation had never before been achieved. To the best of the team's knowledge, there were no real-time project management systems in existence anywhere in the world.

Challenges

This particular undertaking had a number of challenges.

The most important challenge was the resistance to change by client's managers. The new system with its progress and productivity monitoring capabilities threatened to expose inefficiencies and was not universally welcomed. Two approaches were planned to manage this challenge: the first was education of participants and the second, a proposal for a new payment structure which related salaries to meeting of targets, as reported by the new system.

The scale of the proposed network of systems was an even more important challenge, especially because all projects were mission critical. To manage this challenge the plan was made to adopt an evolutionary development strategy. The first step was planned to be a fully engineered prototype with a limited functionality, which would be extended into the first project management system only after a complete acceptance by the client that the prototype was capable of delivering its limited functionality as specified. The network of project management systems would be grown step by step.

The multi-agent technology, which underpinned the system, was well understood by the team, and a methodology for managing complexity (Rzevski and Skobelev, 2014) of the task was in place.

Systems Engineering Practices

Overview

The complexity of client's projects ruled out all conventional project management practices and systems. Instead, for every project, the team designed a complex adaptive project management system, based on multi-agent technology, capable of meeting client requirements.

The system consisted of the following major components (Rzevski and Skobelev, 2014):

1. Knowledge Base containing domain knowledge relevant to the client's project management processes
2. Multi-agent Virtual World which models the Real World of projects and is capable of managing real-world complexity
3. Interfaces between the Virtual and Real Worlds, which enable the Virtual World to, in-effect, manage the Real World, with or without human intervention

Knowledge Base

Examples of Classes of Objects in the ontology are: Enterprise, Organisational Unit, Project, Task, Project Member (Human Resource), Hardware (Physical) Resource, Document, Software Resource, Knowledge Resource and Process.

Examples of attributes are, for Task: Content, Cost, Duration, Deadline and Preferences; and for Project Member: Organisational Unit, Competences, Profile, Schedule, Current Task, Salary, Achievements and Preferences.

A fragment of enterprise ontology is shown below.

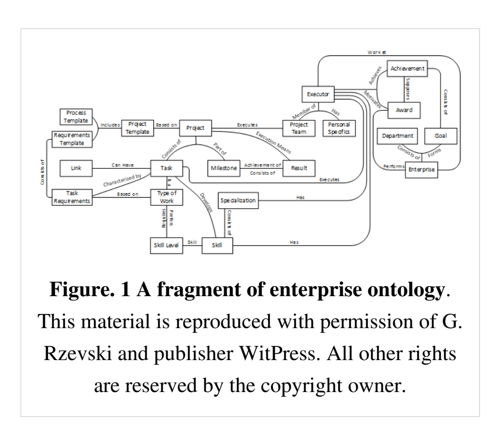


Figure. 1 A fragment of enterprise ontology. This material is reproduced with permission of G. Rzevski and publisher WitPress. All other rights are reserved by the copyright owner.

Virtual World

Examples of agents that populate the Virtual World include:

- Task Agent, whose objective is to search for the best resources capable to meet its requirements
- Human Resource Agent, whose objective is to get the best possible task, which will keep the project participant fully occupied, provide opportunities for bonuses and/or enable the participant to learn new skills or get new experience
- Physical Resource Agent, whose objective is to maximise resource utilisation
- Project Agent, whose objective is to maximise Project Value
- Enterprise Agent, whose objective is to maximise Enterprise Value

All decisions are made through agent negotiations, as exemplified by the following process:

Task Agents send messages to Human Resource Agents with required competences inviting them to contribute to task fulfilments. Human Resource Agents that are available send their bids. Task Agents offer project participation to those Agents that sent the best bids. Bids are subject to negotiations between affected agents.

A new Task Agent is created whenever a new task is formulated or a previously allocated task is modified. The new Task Agent consults ontology to find out what are its objectives and how to achieve them, and proceeds to send messages to selected Human Resource Agents inviting them to bid for project participation. It is very likely that this invitation will result in re-scheduling, giving an opportunity to Human Resource Agents that were not fully satisfied with their previous allocations to improve their positions. Remuneration, including bonuses, if any, is calculated on the basis of project member participation and achieved results. Enterprise members may participate in several projects.

The allocation of physical, software and knowledge resources is done in an analogous manner. Advanced methods (Rzevski and Skobelev, 2014) have been employed to maximise effectiveness of agent negotiation, such as, virtual microeconomics, agent satisfaction, agent proactivity, enterprise agents, swarm cooperation, etc.

Decisions on allocation of resources to project tasks are made using multiple criteria, for example, decreases in completion time, increases in quality and reducing identified risks, as illustrated below.

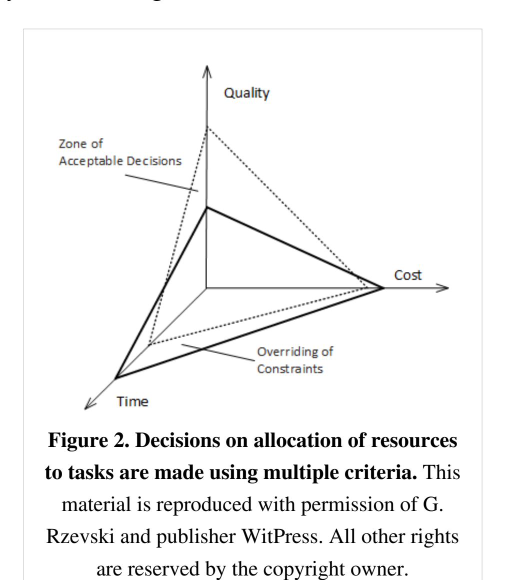


Figure 2. Decisions on allocation of resources to tasks are made using multiple criteria. This material is reproduced with permission of G. Rzevski and publisher WitPress. All other rights are reserved by the copyright owner.

Connecting Virtual and Real Worlds

The project member dashboard is the key link between the system and the project member. The exchange of information that could be conducted using the dashboard include:

- Negotiations of task content, risks, deadlines and budgets
- Acceptance/rejection by the project member of offered project tasks
- Inputs by project members of unexpected disruptive events and comments during the project
- Reports by the system on the project member performance in carrying out accepted tasks

The system may decide to engage two or more available project members in competition with each other to secure an agreement on the acceptable task performance.

The other key link between the virtual and real worlds is the project managers dashboard, which displays detailed project performance data in the form of various diagrams and Gant charts and allows the manager to examine or modify decisions made autonomously by the system.

Lessons Learned

The first real-time complex adaptive project management system was commissioned and deployed by the client in 2014 achieving the following results:

- 10% to 15% increase in project member productivity;
- 3 to 4 times reduction in manpower required for project scheduling, monitoring and coordination;
- 2 to 3 times reduction of response time to unpredictable disruptive events;
- 15% to 30% increase in the number of projects completion on budget and in time;
- A significant increase in project member motivation;
- A possibility to increase the number of projects operating in parallel without increasing the number of employees.

References

Works Cited

Rzevski, G., Skobelev, P., “Managing Complexity” WIT Press, New Forest, Boston, 2014. ISBN 978-1-84564-936-4.

Primary References

Rzevski, G., Skobelev, P., “Managing Complexity” WIT Press, New Forest, Boston, 2014. ISBN 978-1-84564-936-4.

Additional References

None.

Successful Business Transformation within a Russian Information Technology Company

This article describes a successful business transformation of an information technology enterprise. The topic may be of particular interest, especially because this transformation was accomplished by a Russian company during the republic's fast growing economic recovery.

For addition information, refer to the closely related topics of Enabling Businesses and Enterprises and Enterprise Systems Engineering.

Background

In 2001, the top management of the IBS company [1] in Moscow initiated a fundamental transformation to change the company's strategy and business model. The company was one of the biggest Russian information technology (IT) systems integrators at that time, with about 900 employees. Annual revenues of about $80M were mainly generated by information technology (IT) infrastructure projects (complex computing systems, multi-service networks, etc.) and hardware and software distribution. The transformation of the company to form new capabilities in IT services and the associated consulting area is the main topic in the case study.

During the transformation period (from 2001 to the present) IBS was represented as a set of autonomous business units (BUs), called constituent systems, which are virtual, independent businesses with the following characteristics.

- Profit and loss reporting was required for each BU according to management accounting procedures
- BU management established and independently conducted human resources, technology, and product policy
- A centralized back-office was organized to provide supporting functions for each BU. Thus, BUs do not have back-offices; they rely on and "pay" a corporate governing center (CGC) for these services.

A thorough Enterprise System (glossary) (ES) transformation was executed as a set of activities: mission analysis and capabilities decomposition, business architecting, planning of the project program, and implementation of the new business model.

Before and after transformation IBS was an exemplar directed System of Systems (SoS) (glossary): the constituent BUs are autonomous but their operations are supervised by CGC. At the same time IBS also has significant features of an acknowledged SoS: the constituent BUs retain their independent development and sustainment approaches, and changes in the company are based on collaboration between the CGC and each constituent; even operations of BUs are not controlled but only supervised/governed by the CGC through "soft" recommendations and coordination.

IBS was a quite mature ES before the transformation, and it was thoroughly upgraded to form new capabilities of the whole system as well as of the constituents.

Purpose

In 2000-2001 IBS management forecasted considerable growth of the Russian IT services and consulting market based on the fast growing Russian economy, which was rapidly recovering from the national financial crisis of 1998. The largest corporations started overseas expansion and borrowed from international markets to finance this growth. IBS predicted corresponding growth in the complexity of business processes and their associated software and hardware systems all of which should require more consulting and IT services.

Based on this forecast, management established a strategy goal to double the share of IT services and consulting from 25% to 50% over one year; further growth in this business was planned as a long term trend.

The consulting and IT services business is very complex technologically and organizationally and dramatically differs from IBS's former infrastructure focus. Thus, a fundamental transformation was required, and it was executed

during 2002.

Initially detected problems appeared as expenditures exceeding resources, slow delivery of the projects and reworking. Later, as it was expected, new problems appeared, for example, disinterest of BUs' managers in developing new technologies or raising qualified employees' motivation. All those problems were solved during transformation and during further development.

The first step of the transformation included strategic analysis and mission-to-capabilities decomposition. Five major capability groups to be focused on were defined. The groups and exemplar capabilities for each group are represented at Figure 1.

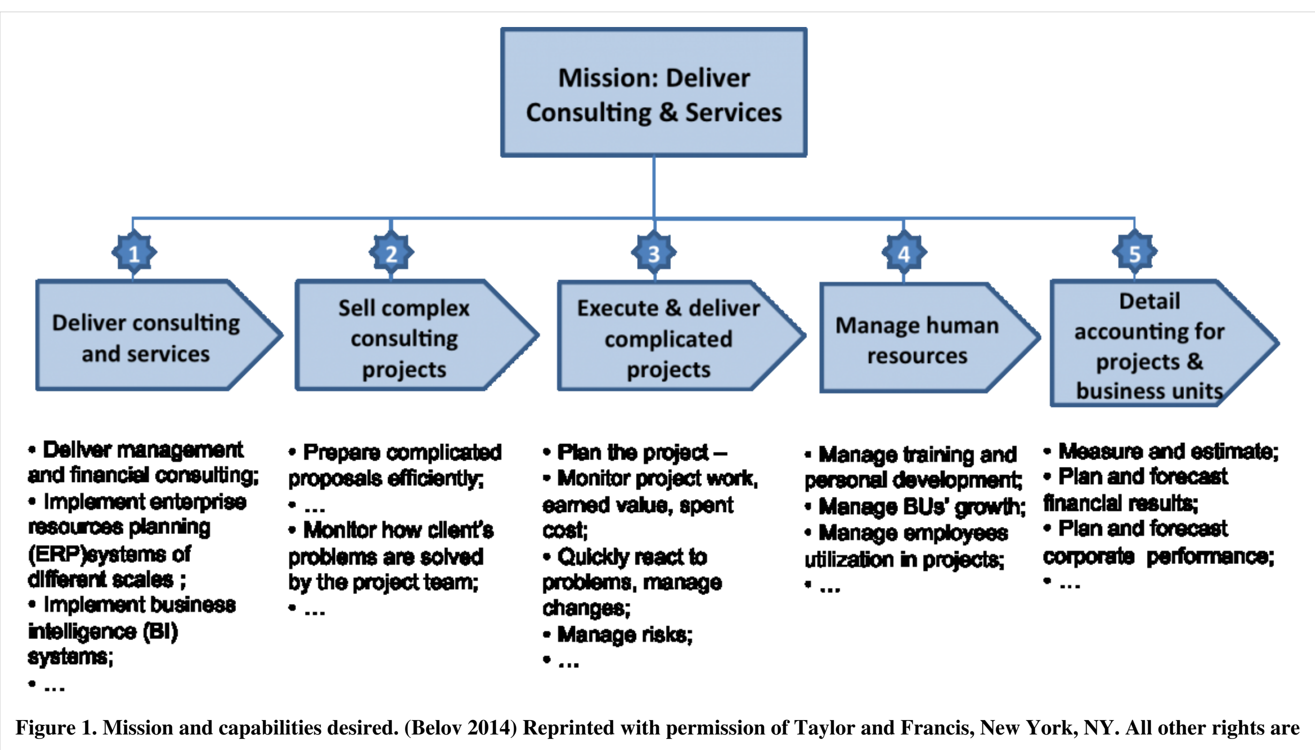


Figure 1. Mission and capabilities desired. (Belov 2014) Reprinted with permission of Taylor and Francis, New York, NY. All other rights are reserved by the copyright owner.

Challenges

All main challenges were caused by knowledge/information deficit described by three factors listed as a, b, and c below.

a. The lack of experience in enterprise transformation (and capability based approaches, even the lack of any textbooks or guides in those areas) was the major challenge which IBS management faced. The task to be solved did not devolve to organizational changes (which was a well-developed and described area), but was appropriately allocated to Enterprise System (glossary) or system of systems (SoS) engineering. In spite of the lack of experience it was decided to prepare and execute the transformation based on the company's employees without involving external consultants. The following arguments supported the decision.

- The task to be solved was not typical, so there weren't widely used and well tested algorithms or methods, and there weren't a lot of consultants experienced in exactly what was needed. So only consultants with general experience (strategy consulting, organizational management) might be hired.
- The Russian consulting industry in 2001-2002 was not well developed, so only foreign professionals were available. But foreign consultants would have needed to study Russian specifics; such study would have unduly lengthened the duration and increased the cost of the transformation.
- A joint transformation team would have to be formed, and IBS employees would have to be involved: management would have to be interviewed and be involved in decision making. In any case all employees would

have to participate in change implementations.

- External consultants are not stakeholders; so their level of interest in helping to achieve success might not be very high, and their output also might not be outstanding.
- Unwillingness to open professional secrets and other intellectual property issues to direct competitors were other factors that prevented hiring of external consultants.

Thus, the final decision was to execute the transformation without involvement of external consulting resources. A special BoU responsible for business processes development was established and an agile (glossary) program management approach was applied to handle challenges and to pursue opportunities as well as to mitigate risks.

b. A very high complexity IBS as an enterprise system or SoS. Management recognized that the company and its environment was very complex, with a lot different agents, many constituents, and countless relationships; and that an enterprise system or SoS might become even more complex after transformation. This complexification happened as the company became an “extended enterprise”, the governing hierarchies weakened, and the demand for more sophisticated relationships increased.

c. The risk of mistaken forecast of IT market development. The expected growth of the consulting and services market might have not happened. In this case the transformation would have been senseless. This challenge generated additional emotional stress for management.

Systems Engineering Practices

The SE task of the transformation was established in the form: to develop required capabilities for an enterprise system or SoS – IBS company. The SE process might be represented by the following specific IBS interpretation of the Vee (V) Model (glossary) (“V model”) with Stages 1 through 7 (Figure 2).

Initially (Stage 1) the mission was translated to capabilities (Figure 1); “understanding the constituent systems (BUs) and their relationships” was executed. The transformation team found that capabilities might not be directly translated to any business-agent. Neither BUs (they serves as resource pools), nor projects (being temporal elements), nor employees (each of them have a finite set of skills, experience, responsibilities, etc.) might realize necessary capabilities.

Realizing this (Stage 2) transformation team defined several key areas (Figure 2) of company’s operations or activities which were supposed to be changed to form new capabilities. Appropriate artifacts (procedures, guides, documents, software systems) to support new capabilities were developed and implemented for each of the areas; these new assets formed exactly the corporate infrastructure of new business model.

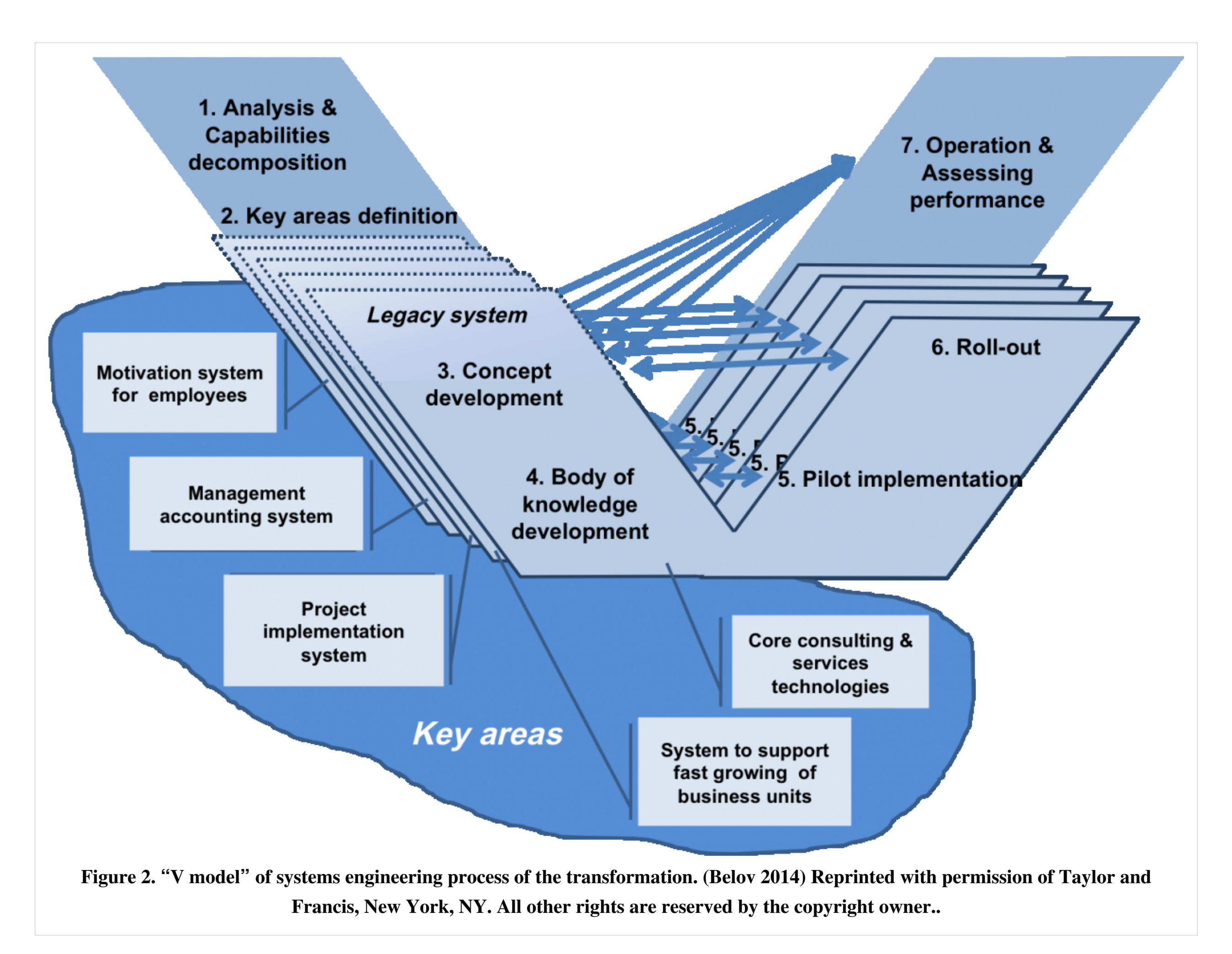


Figure 2. "V model" of systems engineering process of the transformation. (Belov 2014) Reprinted with permission of Taylor and Francis, New York, NY. All other rights are reserved by the copyright owner..

For each new and legacy system (Stage 3) a set of conceptual design documents was developed, describing approaches, polices, processes, and procedures. The entire set of documents formed the business architecture description of the company. The description connected all key areas and defined a target operation model of the company after transformation. This architecture represented multiple views of the IBS company, and thus aptly reflected its enterprise system or SoS nature.

Somewhat in contrast with the conventional linear systems engineering approach advocated by the V model, Stages 4-6 were conducted in parallel to save time and resources. The company's performance (Stage 7) should be monitored based on indicators' measurements, and improvements should be developed and implemented (arrows from Stage 3 to Stage 7). Such iterations have been executed in practice not only during transformation but also later, when procedures, guides and the whole systems were updated.

Integration and interoperability of the new systems required a thorough integration of parallel development jobs. So joint workgroups were formed of the employees at the level of low officers; and CGC played the role of integrated workgroup at the management level. Actually, multi-level integrated workgroups were formed.

The major complexity and risks derived from the challenges described above.

The transformation team developed and used an approach which is very similar to the agile development approach to address those risks. The following principles were used to manage the portfolio of projects in case of uncertainty and deficit of knowledge.

- Form solutions as fast as possible (but not necessarily with pure quality) to test them in practice faster.
- Recognizing failures are unavoidable, perceive them readily and react rationally.
- In case of failure analyze the situation and find a new solution, generate changes, and update the plan.
- Work in parallel, verifying and coordinating intermediate results.

- The schedule might be corrected and updated but should not be jeopardized by improper execution.
- Formulate and test the most critical and most questionable solutions at first.
- Start from pilot area and then expand to embrace the entire scope.
- Use high quality monitoring and a "manual control mode" for piloting and testing developing solutions but not additional aspects to limit waste of the resources.

Following those principles including a very strong discipline of execution, a high level of the sponsorship and all-employee involvement enabled the transformation to be completed on time without hiring consultants while keeping and developing on-going business.

Lessons learned

IBS's accomplishment of the mission was the major result of ES transformation. Shareholders and management recognized that new capabilities had been formed, that the company could deliver consulting and services, sell and execute complex projects, manage consulting resources effectively, measure its performance, and plan and forecast financial results. Created capabilities are emergent in some sense because they are not directly related to concrete constituents (BUs, or employee, or projects) but are realized by means of integrated end-to-end processes and functions, which are executed in the projects by employees.

The systems organization did not dramatically change during transformation; "visible structure" was not practically changed: no new types of business-agents appeared, existing types did not change much. Those factors did not create new capabilities. Target capabilities were formed as the result of development and implementation of, it would seem, auxiliary and supporting tools – new capabilities support systems. New capabilities were formed mainly by the changes in the intangible areas of governing media, corporate culture, relations, and personnel competences; as well as by the creation of new capabilities support systems; without considerable changes in main company's business-agents. (Refer to Figure 3.)

The main challenges which management faced (the lack of experience and the ambiguity of market growth forecast) made the uncertainty factor the critical one in the transformation.

What Worked and Why?

An agile program management in general demonstrated its efficiency and applicability to "soft and uncertain" tasks, especially in triggering a pre-established process for dealing with unexpected events; the main aspects of the approach are:

- Senior and credible sponsors
- Multi-level integrated project team(s)
- Open information exchange
- Partnership and collaboration
- Proactive and motivated parties and constituents
- Creative and innovative way of development

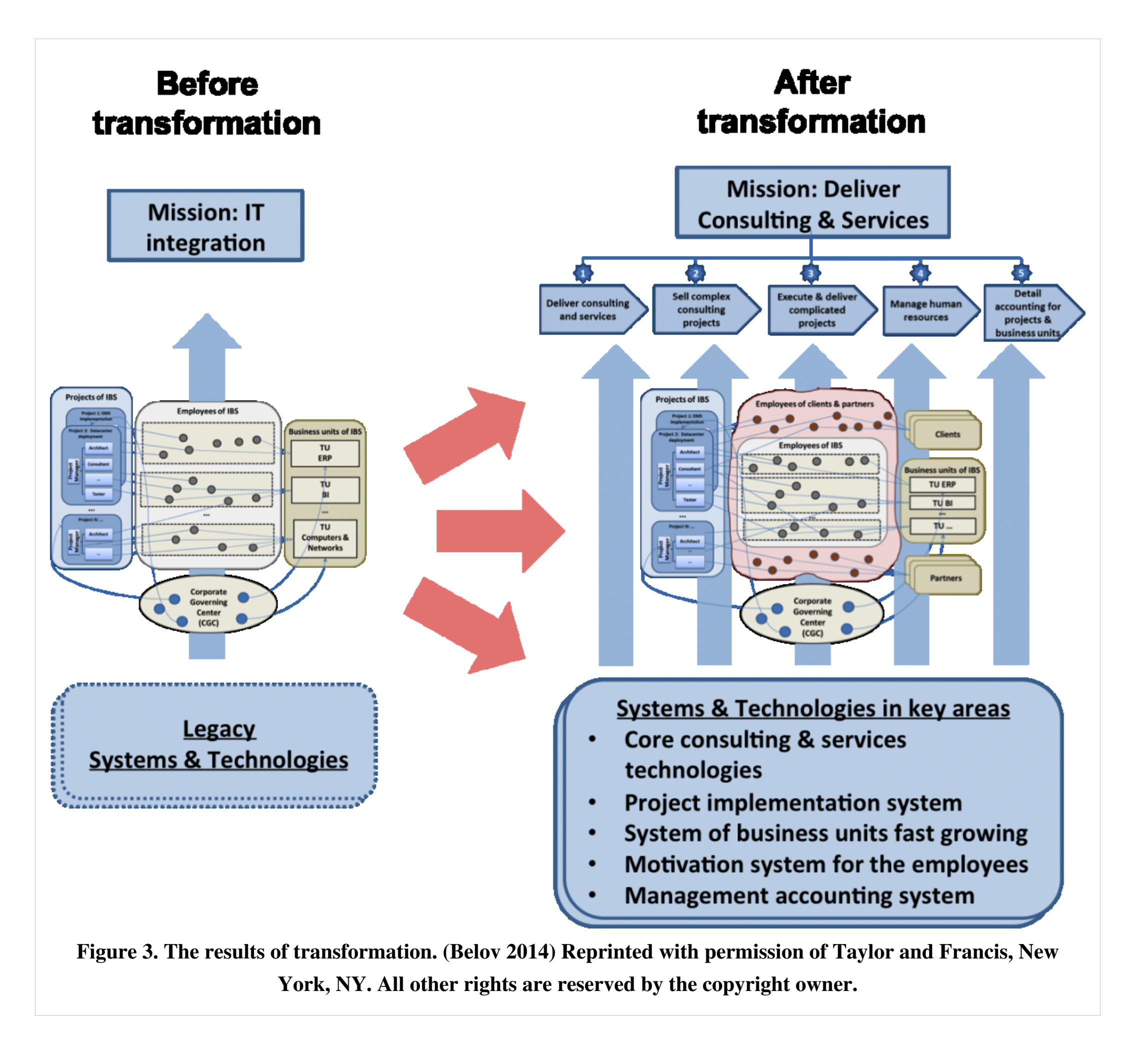


Figure 3. The results of transformation. (Belov 2014) Reprinted with permission of Taylor and Francis, New York, NY. All other rights are reserved by the copyright owner.

- Prioritizing and focusing on the most ambiguous elements of systems design
- Piloting and subsequent roll-out in realistic environments
- Strong project scope control
- Strong project execution control – time schedule and resources control.

What Did Not Work and Why?

Perhaps corporate knowledge base development was the only more or less serious task which was not solved in transformation. The company's management understood the usefulness of knowledge accumulation and further alienation from the carriers in utilizing their business knowledge, so the goal of developing their own knowledge base was established. Special database and software system were developed with appropriate guides, reports and data collection forms; but formal regulation to fill in engineering knowledge accumulation templates did not work. However later this issue progressed quite naturally and simply: common folders were established to store project data in free formats. Such folders served to accumulate knowledge but in flat, unstructured form.

Best Practices and Replication Prospects

The following methods and approaches were proven as efficient and convenient in transformation.

1. Capability based development approach and capability based architecting might be recommended to be utilized in creation and transformation of an enterprise system or SoS. These focused all efforts on the required capabilities and involved very important relations from mission to capabilities and to functions in systems engineering process.
2. An agile program management might be used to solve a wide range of fuzzy and ambiguous problems of different scale in the areas of SE, ES engineering, SoS engineering where there is much uncertainty and lacks of expertise and proven methods or algorithms to solve them. The combination of “soft” and very creative designs with strong planning and progress control is the crucial foundation of this approach.
3. Key area definition and development appropriate to generating new capabilities for support systems (core consulting and services technologies, project implementation systems, systems for business unit growth, management accounting systems, motivation systems). Precisely defining these areas and developing integrated systems in these areas might be considered as quite common for application to a broader group of ESs.

References

Works Cited

Belov, M. 2014. "IBS Group, Eastern European ITS Services – Capability-Based Development for Business Transformation," in *Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering*, edited by A. Gorod et al. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group.

Primary References

Belov, M. 2014. “IBS Group, Eastern European ITS Services – Capability-Based Development for Business Transformation.” Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. Gorod, A., B. E. White, V. Ireland, S. J. Gandhi, and B. J. Sauser. Boca Raton, FL: CRC Press, Taylor & Francis Group. Scheduled for publication in July 2014. http://www.taylorandfrancis.com/books/details/9781466502390/
.

Additional References

None

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.en.ibs.ru/

Government Examples

Federal Bureau of Investigation (FBI) Virtual Case File System

This case study presents systems and software engineering issues encountered in the Federal Bureau of Investigation (FBI) Virtual Case File (VCF) project in the period between 2000-2005. VCF development was abandoned in 2005 after over $170 million had been spent.

Domain Background

The FBI is an organization within the United States Department of Justice (DoJ) consisting of 23 divisions, including counterintelligence, criminal investigation, and cyber crime. The Bureau's 12,400 agents investigate everything from counter-terrorism leads to kidnappings. They interview witnesses, develop informants, conduct surveillance, hunt for clues, and collaborate with local law enforcement to find and arrest criminals. Agents document every step and methodically build case files. They spend a tremendous amount of time processing paperwork. This system of forms and approvals stretches back to the 1920s when forms for all of the bureau's investigative reports were standardized.

In 2000, the Bureau had hundreds of standardized paper forms and obsolete information technology (IT) systems. The FBI’s 13,000 computers could not run modern software. Most of the agency offices were connected to the FBI Intranet with links operating at about the speed of a 56 kilobits-per-second modem. Agents could not e-mail U.S. Attorneys, federal agencies, local law enforcement, or each other; instead, they typically sent case-related information by fax. The agency’s problems in 2000 were summarized in the *9/11 Commission Report*: "the FBI's information systems were woefully inadequate. The FBI lacked the ability to know what it knew; there was no effective mechanism for capturing or sharing its institutional knowledge" (National Commission on Terrorist Acts upon the United States 2004).

In September 2000, Congress approved $380 million over three years for what was then called the FBI Information Technology Upgrade Program. Eventually divided into three parts, the program became known as the Trilogy Information Technology Modernization Program. The first part would provide all 56 FBI field offices with updated computer terminals, as well as new hardware such as scanners, printers, and servers. The second part would re-implement the FBI Intranet to provide secure local area and wide area networks, allowing agents to share information with their supervisors and each other. The third part was intended to replace the FBI's investigative software applications, including the obsolete Automated Case Support (ACS) system.

In June 2001, the FBI awarded a contract to develop the investigative software applications of Trilogy to Science Applications International Corporation (SAIC) over a three year period. The purpose of the software to be developed was to

- provide the capability to find information in FBI databases without having prior knowledge of its location, and to search all FBI databases with a single query through the use of search engines;
- Web-enable the existing investigative applications;
- improve capabilities to share information inside and outside the FBI;
- provide access to authorized information from both internal and external databases; and
- allow the evaluation of cases and crime patterns through the use of commercial and FBI-enhanced analytical and case management tools.

After the September 11 terrorist attacks, the inability of FBI agents to share the most basic information about al Qaeda's U.S. activities was front-page news. Within days, the FBI's obsolete technology infrastructure was being discussed in Congress and the FBI was under intense pressure to improve its information sharing capabilities. On September 4, 2001, Robert S. Mueller III became FBI director, and, in the face of intense public and congressional pressure, Mueller accelerated the Trilogy program. The planned three year period to develop the investigative software was considered politically unacceptable. In January 2002, the FBI requested an additional $70 million to accelerate Trilogy; Congress went further, approving $78 million.

Providing web-enablement of the existing but antiquated and limited ACS system would not provide the investigative case management capabilities required to meet the FBI's post-September 11 mission. In December 2001, the FBI asked SAIC to stop building a Web-based front end for the old programs. Instead, SAIC was asked to devise a new case management system, the Virtual Case File (VCF), to replace ACS. The VCF would contain a major new application, database, and graphical user interface. In order to make both criminal and terrorist investigation information readily accessible throughout the FBI, major changes to the standardized FBI processes would be required. This case study focuses on the VCF component of the Trilogy program.

Case Study Background

The most complete description of the development of the VCF is the report by the DoJ Office of the Inspector General (OIG). The OIG reports to the Attorney General and is independent of the FBI organizations responsible for the Trilogy program. The introduction to the report states, "We conducted this audit to assess the FBI's progress in meeting cost, schedule, technical, and performance targets for the three components of Trilogy. We also examined the extent to which Trilogy will meet the FBI's current and longer-term IT needs" (OIG 2004).

An IEEE *Spectrum* article complements the OIG audit report by providing detailing the development of the VCF requirements, the contractor's activities, and the project management failures by both the FBI and the contractor. The contractor's viewpoint is presented in testimony given before a subcommittee of the U.S. Senate Appropriations Committee.

These materials, in total, provide a comprehensive view of the VCF program and the reasons for its failure.

Case Study Description

In the political environment following the 9/11 attacks, funding for the VCF project was never a problem. By early 2002, SAIC and the FBI committed to creating an entirely new case management system in 22 months. High-level funding enabled the project to continue gaining momentum in spite of the problems it encountered. The scheduling for the VCF project focused on what was desired, not what was possible. Trilogy's scope grew by approximately 80% from the initial project baseline (Moore 2010).

The reasons for the failure of the VCF project are associated with the non-use or misuse of numerous system engineering practices, especially within stakeholder requirements, system requirements, planning, assessment and control, and risk management. Given the political pressures following the 9/11 attacks, the schedule was accelerated to the point that it was nearly impossible for the developers to follow an appropriate systems engineering process.

The FBI cycled through five people in the role of Chief Information Officer in four years and most decisions were made by committees. In order to compress the schedule, the FBI even proposed replacing the ACS with the VCF over a weekend using an IT procedure called a "flash cut-over." In this proposed implementation, the ACS system would be taken offline and entirely replaced by VCF. Once the cut-over happened, there would be no mechanism to return to ACS, even if the VCF did not work properly.

SAIC worked under a cost-plus-award-fee contract for the VCF as the scope of the project was undefined in early 2002 when work began. Given the schedule pressures, the FBI believed that there was no time to develop formal requirements (glossary), validate them with the various FBI user communities, and then estimate the cost and time

required to develop the VCF. The SAIC contract did not require specific completion milestones and the cost-plus contract allowed the scope to increase. VCF was a case of not getting the requirements sufficiently defined in terms of completeness and correctness. The continuous redefinition of requirements had a cascading effect on what had already been designed and produced. Once there was demonstrable software, change requests started arriving—roughly 400 from December 2002 to December 2003.

The new FBI Intranet was specified during 2001, before the start of the VCF project and with little understanding of the network traffic that would result from information sharing. By early 2003, the FBI began to realize how taxing the network traffic would be once all 22,000 users came online. The requirements for the FBI Intranet were modified based on the best guesses for the bandwidth that would be required when the VCF was fully operational. By early 2004, the new FBI Intranet was in operation, although the VCF software was far from complete.

In reaction to the time pressure, SAIC broke its VCF development group into eight teams working in parallel on different functional elements of the program. However, this posed many integration challenges and the eight threads would later prove too difficult for SAIC to combine into a single system. By the time VCF was canceled, SAIC had developed over 700,000 lines of software based upon an incomplete set of requirements that were documented in an 800-page volume.

Summary

The OIG summarizes its conclusions as

> *Various reasons account for the delays and associated cost increases in the Trilogy project, including:*

- *poorly defined and slowly evolving design requirements,*
- *contracting weaknesses,*
- *IT investment management weaknesses,*
- *lack of an Enterprise Architecture,*
- *lack of management continuity and oversight,*
- *unrealistic scheduling of tasks,*
- *lack of adequate project integration, and*
- *inadequate resolution of issues raised in our previous reports on Trilogy. . . .*

> *According to the Government Accountability Office (GAO), an Enterprise Architecture is a set of descriptive models such as diagrams and tables that define, in business and technology terms, how an organization operates today, how it intends to operate in the future, and how it intends to invest in technology to transition from today's operational environment to tomorrow's. . . .*
>
> *As of early 2005 the FBI's operations remain significantly hampered due to the poor functionality and lack of information-sharing capabilities of its current IT systems. . . .* (OIG 2005)

In May 2005, FBI director Mueller announced Sentinel, a four-phase, four-year project intended to fulfill the purpose of VCF and provide the Bureau with a web-based case and records management system. During the previous five years, commercial case management software had become available; as a result, Sentinel is intended to utilize commercial off-the-shelf (COTS) software. A report by the OIG in late 2009 describes Sentinel and its status at that time. Sentinel was put on line for all employees on July 1, 2012, and it ended up at $451 million and 2 1/2 years overdue (Yost 2012).

References

Works Cited

Moore, S. 2010. "The Failure of the FBI's Virtual Case File Project." Strategic PPM: Project and Portfolio Management, last modified April 5, accessed on September 11, 2011. Available at http://strategicppm.wordpress.com/2010/04/05/the-fbis-virtual-case-file-project-and-project-failure.

National Commission on Terrorist Attacks upon the United States. 2004. *The 9/11 Commission Report: Final Report of the National Commission on Terrorist Attacks Upon the United States*. New York, NY, USA: W. W. Norton & Company.

Office of the Inspector General. 2005. *The Federal Bureau of Investigation's Management of the Trilogy Information Technology Project.* Washington, DC, USA: United States Department of Justice. Audit Report 05-07. February 2005. Accessed on September 11, 2011. Available at http://www.justice.gov/oig/reports/FBI/a0507.

Office of the Inspector General. 2009. *Sentinel Audit V: Status of the Federal Bureau of Investigation's Case Management System*. Washington, DC, USA: U.S. Department of Justice. Audit Report 10-03. November 2009. Accessed on September 11, 2011. Available at http://www.justice.gov/oig/reports/FBI/a1003_redacted.pdf.

Yost, P. 2012. "'Sentinel', New FBI Computer System, Finally Tracking Cases -- Years Late and Millions Over Budget." Washington, DC, USA. Accessed on August 6, 2012. Available at http://www.huffingtonpost.com/2012/07/31/sentinel-fbi_n_1725958.html.

Primary References

None.

Additional References

Goldstein, H. 2005. "Who Killed the Virtual Case File?" IEEE *Spectrum*. September. Accessed at September 11, 2011. Available at http://spectrum.ieee.org/computing/software/who-killed-the-virtual-case-file.

Testimony before the Subcommittee on Commerce, Justice, State and the Judiciary, U.S. Senate Committee on Appropriations, February 3, 2005 (statement of Arnold Punaro, Executive Vice President, Science Applications International Corporation).

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Design for Maintainability

This article describes an example of where systems thinking led to a much more practical solution to a common problem. For addition information, refer to Systems Thinking.

This article is excerpted and condensed from Johnson, Steven. 2010. *Where Good Ideas Come From: The Natural History of Innovation*. New York: Riverhead Books. pp. 25-28.

Background

In the late 1870s a Parisian obstetrician named Stephane Tarnier was visiting the Paris Zoo where they had farm animals. While there he conceived the idea of adapting a chicken incubator to use for human newborns, and he hired "the zoo's poultry raiser to construct a device that would perform a similar function for human newborns." At the time infant mortality was staggeringly high "even in a city as sophisticated as Paris. One in five babies died before learning to crawl, and the odds were far worse for premature babies born with low birth weights." Tarnier installed his incubator for newborns at Maternité de Paris, and embarked on a quick study of five hundred babies. "The results shocked the Parisian medical establishment: while 66 percent of low-weight babies died within weeks of birth, only 38 percent died if they were housed in Tarnier's incubating box. … Tarnier's statistical analysis gave newborn incubation the push that it needed: within a few years the Paris municipal board required that incubators be installed in all the city's maternity hospitals." …

Purpose

"Modern incubators, supplemented with high-oxygen therapy and other advances, became standard equipment in all American hospitals after the end on World War II, triggering a spectacular 75 percent decline in infant mortality rates between 1950 and 1998."… "In the developing world, however, the infant mortality story remains bleak. Whereas infant deaths are below ten per thousand births throughout Europe and the United States, over a hundred infants die per thousand (births) in countries like Liberia and Ethiopia, many of them premature babies that would have survived with access to incubators.

Challenges

But modern incubators are complex, expensive things. A standard incubator in an American hospital might cost more than $40,000 [about €30,000]. But the expense is arguably the smaller hurdle to overcome. Complex equipment breaks and when it breaks you need the technical expertise to fix it, and you need replacement parts. In the year that followed the 2004 Indian Ocean tsunami, the Indonesian city of Meulaboh received eight incubators from a range on international relief organizations. By late 2008, when an MIT professor named Timothy Prestero visited the hospital, all eight were out of order, the victims of power surges and tropical humidity, along with the hospital staff's inability to read the English repair manual. The Meulaboh incubators were a representative sample: some studies suggest that as much as 95 percent of medical technology donated to developing countries breaks within the first five years of use.

Systems Engineering Practices

"Prestero had a vested interest in those broken incubators, because the organization he founded, Design that Matters, had been working for several years on a scheme for a more reliable, and less expensive, incubator, one that recognized complex medical technology was likely to have a very different tenure in a developing world context than it would in an American or European hospital. Designing an incubator for a developing country wasn't just a matter of creating something that worked; it was also a matter of designing something that would break in a

non-catastrophic way. You couldn't guarantee a steady supply of spare parts, or trained repair technicians. So instead, Prestero and his team decided to build an incubator out of parts that were already abundant in the developing world. The idea had originated with a Boston doctor named Jonathan Rosen, who had observed that even the smaller towns of the developing world seemed to be able to keep automobiles in working order. The towns might lack air conditioning and laptops and cable television, but they managed to keep their Toyota 4Runners on the road. So Rosen approached Prestero with an idea: What if you made an incubator out of automobile parts?

Lessons Learned

"Three years after Rosen suggested the idea, the Design that Matters team introduced a prototype device called NeoNurture. From the outside, it looked like a streamlined modern incubator, but its guts were automotive. Sealed-beam headlights supplied the crucial warmth; dashboard fans provided filtered air circulation; door chimes sounded alarms. You could power the device via an adapted cigarette lighter, or a standard-issue motorcycle battery. Building the NeoNurture out of car parts was doubly efficient, because it tapped both the local supply of parts themselves and the local knowledge of automobile repair. These were both abundant resources in the developing world context, as Rosen liked to say. You didn't have to be a trained medical technician to fix the NeoNurture; you didn't even have to read the manual. You just needed to know how to replace a broken headlight."

References

Works Cited

Johnson, Steven. 2010. Where Good Ideas Come From: The Natural History of Innovation. New York: Riverhead Books. pp. 25-28.

Primary References

Johnson, Steven. 2010. Where Good Ideas Come From: The Natural History of Innovation. New York: Riverhead Books. pp. 25-28.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Federal Aviation Administration (FAA) Advanced Automation System (AAS)

This example was created as a SE example directly for the SEBoK. It describes the Advanced Automation System (AAS), part of the Federal Aviation Administration (FAA) Advanced Automation Program. It describes some of the problems which can occur in a complex, software intensive system program if SE is not applied.

Description

In 1981 the Federal Aviation Administration (FAA) announced the Advanced Automation Program, which was established to modernize air traffic control (ATC) computer systems. A centerpiece of the project was the Advanced Automation System (AAS). AAS was the largest project in FAA's history to modernize the nation's ATC system. AAS would replace computer hardware and software as well as controller work stations at tower, terminal, and en-route facilities and allow the ATC system to accommodate forecasted large increases in traffic through the use of modern equipment and advanced software functions. (GAO 1992)

The FAA originally proposed AAS in 1982 as a project that would cost $2.5 billion and be completed in 1996. However, substantial cost increases and schedule delays beset the AAS project over it history, caused by numerous problems in AAS development:

- The project began with a design competition between Hughes and IBM. The competition involved numerous extensions and took four years to complete. Analysis by the FAA and others pointed to inadequate consideration of user expectations and improper assessment of the technology risks. (Barlas 1996)
- The FAA pushed for 99.99999% reliability, which was considered by some "more stringent than on any system that has ever been implemented" and extremely costly. (DOT 1998)
- The program created unworkable software testing schedules - "Testing milestones were skipped or shortcutted and new software was developed assuming that the previously developed software had been tested and performed." (Barlas 1996)
- There were an extraordinary number of requirements changes. For example, for the Initial Sector Suite System (ISSS), a key component of AAS, there were over 500 requirements changes in 1990. Because of these changes, 150,000 lines of software code had to be rewritten at a cost of $242 million. (Boppana et al. 2006)
- IBM's cost estimation and development process tracking used inappropriate data, were performed inconsistently, and were routinely ignored by project managers. The FAA conservatively expected to pay about $500 per line of computer code - five times the industry average. The FAA ended up paying $700 to $900 per line for the AAS software. (Gibbs 1994)
- In 1988, FAA estimated that the AAS program - both contract and supporting efforts - would cost $4.8 billion. By late 1993, the FAA estimated that it would cost $5.9 billion. Before the program was dramatically restructured in 1994, estimates had risen to as much as $7 billion, with key segments expected to be behind schedule by as much 8 years. In 1994, with significant cost and schedule overruns, as well as concerns about adequate quality, usability, and reliability, the AAS program ceased to exist as originally conceived, leaving its various elements terminated, restructured, or as parts of smaller programs. (DOT 1998)

The AAS problems could be associated with the non-use or misuse of a number of systems engineering (SE) concepts and practices: system requirements, system architecture complexity, project planning, risk management, change management, system analysis and design, system reliability, system integration, system verification and system validation/testing, and management oversight.

Summary

The AAS program was the centerpiece of an ambitious effort begun in the 1980s to replace the computer hardware and software throughout the ATC system - including controller workstations, and en-route, terminal, and tower air traffic control facilities. AAS was intended to provide new automated capabilities to accommodate increases in air traffic. After sustaining serious cost and schedule problems, FAA dramatically restructured the program into more manageable pieces. This action included terminating major segments of the contract. (DOT 1998)

References

Works Cited

Barlas, S. “Anatomy of a Runaway: What Grounded the AAS.” IEEE *Software*. 13(1): 104-106.

Boppana, K., S. Chow, O.L. de Weck, C. LaFon, S.D. Lekkakos, J. Lyneis, M. Rinaldi, Z. Wang, P. Wheeler, M. Zborovskiy. 2006. "Can Models Capture the Complexity of the Systems Engineering Process?" Proceedings of the International Conference on Complex Systems (ICC2006), 11-15 June 2006, Istanbul, Turkey.

DOT. 1998. *Audit Report: Advance Automation System, Federal Aviation Administration*. Washington, DC, USA: Office of Inspector General, U.S. Department of Transportation.

GAO. 1992. *Advanced Automation System Still Vulnerable to Cost and Schedule Problems*. Washington, DC, USA: United States General Accounting Office (GAO). GAO/RCED-92-264.

Gibbs, W.W. “Software’s Chronic Crisis.” *Scientific American*. September 1994.

Primary References

None.

Additional References

Britcher, Robert. 1999. *The Limits of Software: People, Projects, and Perspectives*. Boston: Addison Wesley.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Federal Aviation Administration (FAA) Next Generation Air Transportation System

This article describes a massive undertaking to modernize the air traffic management enterprise (glossary). The topic may be of particular interest to those involved in air transportation whether in connection with their careers or as pilots or passengers on airplanes. For addition information, refer to the closely related topics of Enabling Businesses and Enterprises and Enterprise Systems Engineering.

Background

This case study presents the Systems Engineering (glossary) and Enterprise Systems Engineering (ESE) (glossary) efforts in the Next Generation (NextGen) Air Transportation Systems by the Federal Aviation Administration (FAA 2008). NextGen is an unprecedented effort by multiple U.S. federal organizations to transform the U.S. air transportation infrastructure (glossary) from a fragmented ground-based navigation system to a net-centric satellite-based navigation system. This project is unique to the FAA because of its large scale, the huge number of stakeholder(s) involved, the properties of the system of interest, and the revolutionary changes required in the U.S. Air Transportation Network (U.S. ATN) enterprise.

A Sociotechnical System (glossary) like the U.S. ATN is a “large-scale [system] in which humans and technical constituents are interacting, adapting, learning, and coevolving. In [such] systems technical constraints and social and behavioral complexity are of essential essence”. (Darabi and Mansouri 2014). Therefore, in order to understand changes in the U.S. ATN it was seen as necessary to view it through a lens of evolutionary adaptation rather than rigid systems design. The U.S. ATN serves both military and commercial aircraft with its 19,782 airports, including 547 are commercial airports. Nineteen major airlines, with more than a billion dollars in annual total revenue, along with other 57 national and regional airlines, transport 793 million passengers and realize 53 billion revenue ton-miles.

The Air Traffic Organization (ATO) is responsible for ensuring aircraft navigation in the U.S. National Air Space (NAS) system using a five-layer architecture (glossary). Each aircraft goes through different layers and possibly various zones of this architecture as it takes off from an airport until its lands at another airport (Donohue and Zellweger 2001). However, this architecture is fragmented and many issues are raised: an airplane’s path through its route is not optimized, and the path may change its direction from one zone to another, the destination airport’s capacity is limited by the current regulations of minimum aircraft separation distance due to navigation limitations, the real-time weather information is not integrated into the system, and communications are mainly voice-based, etc.

In NextGen major changes to the U.S. ATN design are planned. As already stated, the navigation system will be changed from ground-based communication to satellite-based navigation. The current fragmented architecture will be integrated into a seamless net-centric information system in which the digital communication will replace the current voice communications. Moreover, weather information will be assimilated into decision making and planning across the system.

Purpose

The FAA’s purpose is “to provide the safest, most efficient aerospace system in the world”. Toward this end the NextGen project is aimed at enhancing the U.S.’s leadership position in air transportation.

During the last three decades the demand for air transportation shows exponential growth. In just one decade from 1995 to 2005 this demand showed a 44% percent increase. Therefore, the change in infrastructure was inevitable. Moreover, 9/11 attacks on the U.S. ATN emphasized this need for change. The combination of a requirement for a safer and more secure network and increasing demand was the motivation for President Bush to enact the Vision 100-Century of Aviation Reauthorization Act on 2003. A major part of this Act was to revolutionize the U.S. ATN

by means of the NextGen project. The first integration plan of the project was released in 2004, and the project is estimated to continue until 2025.

The demand behavior of the U.S. ATN shows diverse degrees of congestion among airports. Although there are multitudes of airports in the system, the top 35 most congested airports carried more than 60% of the total traffic consistently during the period of 2000 to 2008. Because the growth of the network demand is not proportional, the demand in congested airports will be even higher.

A study by the Joint Planning and Development Office (JPDO) shows that flight delays in the current network will cause $6.5 billion of economic loss until 2015, and $19.6 billion until 2025. By implementing NextGen the delays are estimated to be reduced by 38% until 2020. Moreover, aircraft CO2 emissions are a major part of environmental pollution in crowded cities; these will be reduced by 14 million metric tons by 2020. The current level of jet fuel usage is also a known problem because of increasing fuel prices. The NextGen project will improve fuel usage by 1.4 billion gallons cumulative through 2020.

NextGen is pursuing multiple goals to retain the U.S. leadership in aviation, to expand the U.S. ATN capacity, to continue to ensure safety, to increase environment protection, to help ensure national air defense, all generally helping to increase the nation’s security (JPDO 2007a).

Eight general capabilities are defined in conducting this mission: (1) network-enabled information access, (2) performance-based operations and services, (3) weather assimilated into decision making, (4) layered adaptive security, (5) positioning, navigation, and timing (PNT) services, (6) aircraft trajectory-based operations (TBO), (7) equivalent visual operations (EVO), and (8) super-density arrival/departure operations.

To create the desired capabilities, general areas of transformations are defined as air traffic management operations, airport operations and infrastructure services, net-centric infrastructure services, shared situational awareness services, layered and adaptive security services, environmental management services, safety management services, and performance management services. The detailed changes in each area are discussed in Concept of Operations for NextGen (JPDO 2007a).

Challenges

An instructive part of this case study is observing evolution in understanding challenges from initial steps of the project through current efforts for delivering it. As an overall conclusion, the perspective on challenges shifted from technical problems and intra-organizational issues to more enterprise-wide issues.

The NextGen Implementation Plan 2008 discussed the following challenges (FAA 2008):

- performance analysis, to understand and assess operational capabilities
- policy, to balance responsibility between humans and automation, for environmental management processes, and for global harmonization strategies
- acquisition workforce staffing
- environmental planning, to resolve conflicts with local environmental constraints
- security
- transition from current ground-based navigation to automatic dependent surveillance – broadcast (ADS-B) technology.

A more recent report on Targeted NextGen Capabilities for 2025 (JPDO 2011) highlights the effect of the multi-stakeholder nature of the project on raising additional challenges. Achieving Interagency Collaboration is the first issue, which is important in implementing security, safety, policy making, and technological advancement.

Increasing capacity, reducing delay and protecting the environment are the main three promises of the NextGen project. However, reaching the defined high standards is not an easy task. A major part of this challenge is integrating new technologies into legacy systems, aircraft, airports, facilities, and organizations. Airlines and general aviation pilots resist the expense of additional avionics and communications equipment, even though it bolsters the

common good of air travel.

Maintaining airports and airspace security requires coherent and harmonious work of multiple U.S. agencies. The core of this challenge is not just changing the technology but also the processes, organizational structures, and enterprises to meet the new requirements of security.

Moreover, the need for greater information sharing in this net-centric environment is a challenge. The current culture of limited information sharing in which inter-organizational and intra-organizational information is strictly divided creates tension in a seamless information sharing infrastructure. In addition to that, the responsibility of generating, sharing, and utilizing useful information should be addressed in advance to avoid costly mistakes.

Verification and validation of NextGen deliverables is a major issue. The traditional systems engineering methods of verification and validation are tailored for testing an isolated system, while by definition a project like NextGen requires new methodologies of verification and validation beyond the scope of one system. The knowledge and experience of advancement in systems engineering in this area can be of priceless value for future projects.

Balance between human decision-making and automation is required to ensure a correct policy for increasing traffic and safety concerns. Changes in both human resource and technological facilities are required to effectively address this challenge.

The support of local communities is essential to facilitate development of the U.S. ATN and its physical infrastructure.

Communication, navigation, and surveillance systems in NextGen are going through major changes in terms of capacity and technology. However, planning required backups for them in case of any emergency is an area of challenge in developing NextGen.

The rise of Unmanned Aircraft Systems (UASs) provides significant opportunities for both military and commercial applications. However, integrating them into the NAS and developing policing and strategies for safe and secure use is a concern for the revolutionized U.S. ATN.

And finally realizing the benefits of NextGen is dependent on the critical mass of early adopters, similar to any technological advancement. Therefore, the NextGen project authority requires well-defined policies for motivating stakeholders' participation.

Systems Engineering Practices

The FAA NextGen is not just a revolution of the U.S. air transportation infrastructure, but also a shift in its enterprise. The enterprise architecture document, which is developed by JPDO, provides an overview of the desired capabilities (JPDO 2007b).

The Enterprise Architecture (glossary) is described using Department of Defense Architecture Framework (DoDAF) and the Federal Enterprise Architecture (FEA). DoDAF is used to describe the operational aspects of the project. The three views of DoDAF, the Operational View (OV), the Systems View (SV), and the Technical Standards View (TV), are presented in the enterprise architecture document. The Overview and Summary Information (AV-1) is the formal statement about how to use the architecture, the Integrated Dictionary (AV-2) defines the terms in the document, the Community Model (OV-1) presents a high level depiction of the NextGen community, the Operational Node Connectivity Description (OV-2) presents the information flow among operational nodes in the system, Operational Information Exchange Matrix (OV-3) details the description of information flow in OV-2. Other architectural views of the system based on DoDAF are the Activity Model (OV-5) which documents activities (functions and processes), the Operational Event/Trace Description (OV-6c) is a part of sequence and timing description of activities, the System Functionality Description (SV-4) explains system functional hierarchies, and the Operational Activity to System Functionality Traceability Matrix (SV-5) is specification of relationships between operational activities in architecture and functional activities. However, a challenging part of applying this Enterprise Architecture is transformation from legacy systems to the new NextGen. This transformation is the ultimate test for

relevance and comprehensiveness of the developed Enterprise Architecture.

Acquisition is the heart of systems engineering activities in the FAA NextGen project. As mentioned in Challenges above, the current practice of verification of validation in systems engineering (SE) is geared toward single isolated systems, rather than a myriad of interconnected System of Systems (SoS) (glossary). Moreover, the capabilities of NextGen are interdependent, and different programs rely on each other to deliver the promises. 250 unique and highly interconnected acquisition programs are identified in the FAA's Capital Investment Plan, and these are to be delivered by 1820 FAA acquisition professionals. In addition, program complexity, budget uncertainty, and the challenge of finding acquisition professionals present other problems. The experience of systems acquisition in NextGen can provide a useful knowledge for future similar projects.

Lessons Learned

Although major portions of the FAA NextGen project are technical transformations and physical infrastructure developments, the transformation in the aviation enterprise is important but to some degree neglected. Part of the issue might be the fact that this transformation is beyond the responsibility and capability of FAA. However, to accomplish NextGen's perceived benefits it is important to realize the effects of legacy systems, and most importantly the legacy enterprise architecture of the U.S. ATN. Many of the actual challenges in the system arose because of this inattention.

The sequestration in the U.S. government, the Budget Control Act of 2011, has cut the project funding substantially in recent years. As a result the project schedule and portfolio are subject to constant and wide-spread changes. The FAA was focused on delivering Optimization of Airspace and Procedures in the Metroplex (OPAM) program which is designed to reduce the delay, fuel consumption, and exhaust emission in busiest airports. The three areas of Houston, North Texas, and Washington D.C. were planned to complete the design phase on 2013 and start implementation.

Out of 700 planned ADS-B ground stations, 445 were operational on February 2013. ADS-B capability is a NextGen descendant of current radar systems and provides situational awareness for the players in the NAS using the Global Positioning System (GPS) and Wide Area Augmentation System (WAAS).

On the enterprise part of the project, the FAA Modernization and Reform Act of 2012 provided financial incentives for airlines and commercial aviation manufacturers to implement the required equipment in their aircraft. These incentives are designed to engage the air transportation community in the project and to create the critical mass of equipped airplanes.

There are considerable practices in applying NextGen. Establishment of the JPDO made the efforts of the project more coherent and integrated. JPDO's main responsibility is to coordinate development of NextGen. The role of this organization is to represent multiple stakeholders of the project, which enables it to resolve possible conflicts of interests inside one entity. Moreover, such an organization provides a venue for technical knowledge-sharing, creating a consensus, and making an integrated system.

Emphasizing delivery of the mid-term objectives of NextGen is another lesson of the project. It was a well-known practice documented by Forman and Maier to establish mid-points for complex projects (Forman 2000). Developing a mid-level system provides the system designers an opportunity to examine their underlying assumptions, to identify best practices and heuristics in the context of the project, and to reapply the acquired knowledge thorough evolutionary developments. A major shift in the policy of FAA in recent years was to focus on delivering project mid-term objectives.

There are unique characteristics of NextGen which makes it a valuable case for learning and replicating to other complex transformation projects of sociotechnical systems. The scale of the project for infrastructure transformation is unprecedented. The system includes legacy systems and cutting edge technology, and its performance is based on their coherent work. The project implementation is dependent on involved participation of multiple governmental and commercial organizations. Moreover, this case-study provides a great investigation in enterprise governance and

enterprise transformation beyond a single organization.

References

Works Cited

Darabi, H.R., and M. Mansouri. 2014. "NextGen: Enterprise Transformation of the United States Air Transport Network," in *Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering*, edited by A. Gorod et al. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group.

Donohue, G.L., and A.G. Zellweger. 2001. "Air Transportation Systems Engineering." *American Institute of Aeronautics and Astronautics*. 193 (1).

FAA. 2008. *FAA's NextGen Implementation Plan. Federal Aviation Administration*. Washington, DC, USA. Accessed March 29, 2014. Available: http://www.faa.gov/nextgen/media/ng2008_implementation_plan.pdf.

Forman, B. 2000. "The political process and systems architecting," in *The Art of Systems Architecting*, 2nd ed., edited by M.W. Maier and E. Rechtin. Boca Raton, FL, USA: CRC Press LLC.

JPDO. 2007a. *Concept of operations for the next generation air transportation system.* Joint Planning Development Office. Accessed March 29, 2014. Available: http://www.jpdo.gov/library/nextgen_v2.0.pdf.

JPDO. 2007b. *Enterprise Architecture V2.0 for the Next Generation Air Transportation System.* Joint Planning and Development Office. Accessed March 29, 2014. Available: http:/ / www. jpdo. gov/ library/ EnterpriseArchitectureV2.zip.

JPDO. 2011. *Targeted NextGen Capabilities for 2025*. Joint Planning and Development Office. Accessed March 29, 2014. Available: http://www.jpdo.gov/library/2011_Targeted_NextGen-Capabilities_for_2025_v3.25.pdf.

Primary References

Darabi, H.R. and M. Mansouri. 2014. "NextGen: Enterprise Transformation of the United States Air Transport Network." Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. Gorod, A., B.E. White, V. Ireland, S. J. Gandhi, and B.J. Sauser. Boca Raton, FL: CRC Press, Taylor & Francis Group. Scheduled for publication in July 2014. http:/ / www. taylorandfrancis. com/ books/ details/ 9781466502390/ . Accessed 29 March 2014.

Additional References

None.

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn

This article describes a deep space mission where more forthright information exchanges between teamed but rival agencies could have preserved the original plan as well as saved much time and money. The topic may be of particular interest to those involved in institutional collaborations where there are vested interests in protecting rather than sharing information.

For addition information, refer to the closely related topics of Information Management, Organizing Business and Enterprises to Perform Systems Engineering and Fundamentals of Services.

Background

Before the "Faster, Better, Cheaper" philosophy introduced in the 1990s, the United States National Aeronautics and Space Administration (NASA) focused on three classes of unmanned space missions. In order of increasing cost, these were the Discovery, New Frontiers, and Flagship programs. Flagship programs typically cost more than $1B, and included the Voyager (outer planets), Galileo (Jupiter), Cassini-Huygens (Saturn), Mars Science Laboratory (Mars), and the James Webb Space Telescope. (Wall 2012)

The concept of the Cassini-Huygens mission was initiated in 1982 as the result of a working group formed by the National Academy of Sciences and the European Science Foundation. This group sought opportunities for joint space missions; several subsequent reports endorsed the working group's concept of a Saturn orbiter coupled with a Titan (Saturn's largest moon) lander. (Russell 2003, p. 61)

By 1988, NASA was politically motivated to reverse earlier tensions with the European Space Agency (ESA) by engaging in a joint mission. Cassini-Huygens was seen as a mechanism to achieve this goal, and the cooperation between NASA and ESA helped the program survive potential budget cuts (since the U.S. was obligated to match ESA commitments). (Russell 2003, p. 62)

NASA and ESA approved the Cassini-Huygens program, and it proceeded under a traditional management approach. NASA built the Cassini orbiter (the largest and most complex unmanned space probe ever built) and the ESA constructed the Huygens lander. This partition of responsibility almost led to the failure (glossary) of the Titan survey portion of the mission. Cassini (which would conduct a variety of scientific surveys of the Saturn planetary system) was expected to relay transmissions from Huygens to NASA's Deep Space Network (DSN); however, the interface between the lander and orbiter was not well-managed and erroneous assumptions about how the orbiter/lander system would behave after separation nearly doomed the Titan exploration portion of the mission. (Oberg 2004)

Purpose

The intent of the Titan survey portion of the Cassini-Huygens mission was that the Huygens lander would separate from the Cassini orbiter and commence a one-way, 2.5 hour descent into Titan's atmosphere. Its modest transmitter would send data back to the orbiter, which would relay the information to Earth. (Oberg 2004, p. 30) This effectively made the radio link between the two spacecraft a single point of failure (SPOF) and one that was not well characterized.

Alenia Spazio SpA, the Italian communications vendor that built the radio system, overlooked the Doppler shift (approximately 38 kHz) (Oberg, 2004, p. 31) that would occur when Huygens separated from Cassini and began its descent (Oberg 2004, p. 38). The communications protocol was binary phase-key shifting: "[the] transmission system represents 1s and 0s by varying the phase of the outgoing carrier wave. Recovering these bits requires precise timing: in simple terms, Cassini's receiver is designed to break the incoming signal into 8192 chunks every second. It determines the phase of each chunk compared with an unmodulated wave and outputs a 0 or a 1 accordingly". (Oberg 2004, p. 31) The receiver was appropriately configured to compensate for the Doppler shift of the carrier wave but would be unable to adjust for the Doppler shift of the encoded data. "In effect, the shift would push the signal out of synch with the timing scheme used to recover data from the phase-modulated carrier." (Oberg 2004, p. 33) Therefore, the communications system would be unable to decode the data from the lander and would then relay scrambled information to NASA. Because of the failure mechanism involved, the data would be completely unrecoverable.

Both Cassini and Huygens had been tested before launch; however, none of the testing accurately reflected the Doppler shift that would be experienced at this critical phase of the mission. An opportunity to conduct a full-scale, high-fidelity radio test was ignored due to budget constraints; the testing would have required disassembly and subsequent recertification of the probes. (Oberg, 2004, p. 30) Correcting this latent issue would have been trivial before the spacecraft were launched (via a minor firmware upgrade); (Oberg 2004, p. 33) once they were on the way to Saturn any corrective action would be severely limited and expensive.

Once the mission was underway, the probe coasted along its seven-year trajectory to Saturn and its moons. Claudio Sollazzo, the ESA ground operations manager, was uncomfortable with the untested communications system. He tasked Boris Smeds, an engineer with radio and telemetry experience, with finding a way to test the communications system using an Earth-generated signal. (Oberg 2004, p. 30)

Smeds spent six months developing the test protocols that would use Jet Propulsion Laboratory (JPL) ground stations and an exact duplicate of Huygens. Simulated telemetry would be broadcast from Earth to Cassini and relayed back; the test signal would vary in power level and Doppler shift to fully exercise the communications link and accurately reflect the anticipated parameters during Huygens's descent into Titan's atmosphere. (ESA 2005)

Challenges

Smeds faced opposition to his test plans from those who felt it was unnecessary, but ultimately prevailed due to support from Sollazzo and Jean-Pierre Lebreton, the Huygens project (glossary) scientist. More than two years after the mission was launched, Smeds traveled to a DSN site in California to conduct the test. (Oberg 2004, p. 31)

A test signal was broadcast, received by Cassini, re-transmitted to the DSN site, and relayed to ESA's European Space Operation Centre (ESOC) in Darmstadt, Germany for analysis. Testing had to be conducted when the orbiter was in the correct relative position in the sky; it was more than a quarter of a million miles away with a signal round-trip time of nearly an hour. The test immediately exposed an issue; the data stream was intermittently corrupted, with failures not correlated to the power level of the test signal. The first of two days of testing concluded with no clear root cause identified. (Oberg 2004, p. 31)

Even though the probe was far from its ultimate destination, many science teams were competing for time to communicate with it using the limited bandwidth available. The communications team would not be able to conduct

another set of trials for several months. Smeds diagnosed the root cause of the problem; he felt it was the Doppler shifts induced in the simulated signal. However, the test plan did not include unshifted telemetry (an ironic oversight). He modified his test plan overnight and shortened the planned tests by 60%; this recovered sufficient time for him to inject an unshifted signal into the test protocols. (Oberg 2004, p. 32)

This unshifted signal did not suffer from the same degradation; however, other engineers resisted the diagnosis of the problem. Follow-up testing using probe mockups and other equipment ultimately convinced the ESA of the issue; this took an additional seven months. (Oberg 2004, p. 33)

By late 2000, ESA informed NASA of the latent failure of the communications link between Cassini and Huygens. Inquiry boards confirmed that Alenia Spazio had reused timing features of a communications system used on Earth-orbiting satellites (which did not have to compensate for Doppler shifts of this magnitude). (Oberg, 2004, p. 33) In addition, because NASA was considered a competitor, full specifications for the communications modules were not shared with JPL. The implementation of the communications protocols was in the system's firmware; trivial to correct before launch, impossible to correct after. (ESA 2005)

A 40-man Huygens Recovery Task Force (HRTF) was created in early 2001 to investigate potential mitigation actions. Analysis showed that no amount of modification to the signal would prevent degradation; the team (glossary) ultimately proposed changing the trajectory of Cassini to reduce the Doppler shift. (ESA 2005) Multiple studies were conducted to verify the efficacy of this remedy, and it ultimately allowed the mission to successfully complete the Titan survey.

Systems Engineering Practices

Space missions are particularly challenging; once the spacecraft is en route to its destination, it is completely isolated. No additional resources can be provided and repair (particularly for unmanned mission) can be impossible. Apollo 13's crew barely survived the notable mishap on its mission because of the resources of the docked Lunar Excursion Module (LEM) and the resourcefulness of the ground control team's experts. A less well-known failure occurred during the Galileo mission to Jupiter. After the Challenger disaster, NASA adopted safety standards that restricted the size of boosters carried in the Space Shuttle. (Renzetti 1995) Galileo was delayed while the Shuttles were grounded and Galileo's trajectory was re-planned to include a Venus fly-by to accelerate and compensate for a smaller booster. Galileo's main antenna failed to deploy; lubricant had evaporated during the extended unplanned storage (Evans 2003) and limited computer space led to the deletion of the antenna motor-reversing software to make room for thermal protection routines. When the antenna partially deployed, it was stuck in place with no way to re-furl and redeploy it. Engineers ultimately used an onboard tape recorder, revised transmission protocols, the available low-gain antenna, and ground-based upgrades to the DSN to save the mission. (Taylor, Cheung, and Seo 2002)

The Titan survey was ultimately successful because simulation (glossary) techniques were able to verify the planned trajectory modifications and sufficient reaction mass was available to complete the necessary maneuvers. In addition, Smeds's analysis gave the mission team the time it needed to fully diagnose the problem and develop and implement the remedy. If this test were conducted the day before the survey it would merely have given NASA and ESA advance warning of a disaster. The time provided enabled the mission planners to craft a trajectory that resolved the communication issue and then blended back into the original mission profile to preserve the balance of the Saturn fly-bys planned for Cassini. (Oberg 2004, p. 33)

Lessons Learned

The near-failure of the Cassini-Huygens survey of Titan was averted because a handful of dedicated systems engineers fought for and conducted relevant testing, exposed a latent defect, and did so early enough in the mission to allow for a recovery (glossary) plan to be developed and executed. Root causes of the issue included politically-driven partitioning, poor interface management, overlooked contextual information, and a lack of appreciation for single-points-of-failure (SPOFs).

The desire to use a joint space mission as a mechanism for bringing NASA and ESA closer together (with the associated positive impact in foreign relations) introduced an unnecessary interface into the system. Interfaces must always be managed carefully; interfaces between organizations (particularly those that cross organizational or political borders) require extra effort and attention. Boeing and Airbus experienced similar issues during the development of the Boeing 787 and A380; international interfaces in the design (glossary) activities and supply chains led to issues:

> *…every interface in nature has a surface energy. Creating a new surface (e.g., by cutting a block of steel into two pieces) consumes energy that is then bound up in that surface (or interface). Interfaces in human systems (or organizations), a critical aspect of complex systems such as these, also have costs in the effort to create and maintain them. Second, friction reduces performance. Carl von Clausewitz, the noted military strategist, defined friction as the disparity between the ideal performance of units, organizations, or systems, and their actual performance in real-world scenarios. One of the primary causes of friction is ambiguous or unclear information. Partitioning any system introduces friction at the interface.* (Vinarcik 2014, p. 697)

Alenia Spazio SpA's unclear understanding of the Doppler shift introduced by the planned relative trajectories of Huygens and Cassini during the Titan survey led it to reuse a component from Earth-orbiting satellites. Because it considered NASA a competitor and cloaked details of the communications system behind a veil of propriety, it prevented detection of this flaw in the design phase. (Oberg 2004, p. 33)

Because NASA and ESA did not identify this communication link as a critical SPOF, they both sacrificed pre-launch testing on the altar of expediency and cost-savings. This prevented detection and correction of the flaw before the mission was dispatched to Saturn. The resource cost of the later analysis and remedial action was non-trivial and if sufficient time and reaction mass had not been available the mission would have been compromised. It should be noted that a number of recent spacecraft failures are directly attributable to SPOFs (notably, the Mars Polar Lander (JPL 2000) and the Genesis sample return mission (GENESIS, 2005)). Effective SPOF detection and remediation must be a priority for any product development effort. More generally, early in the development process, significant emphasis should be placed on analyses focused on what might go wrong ("rainy day scenarios") in addition to what is expected to go right ("sunny day scenarios").

The success of the Huygens survey of Titan was built upon the foundation established by Boris Smeds by identifying the root cause of the design flaws in a critical communications link. This case study underscores the need for clear contextual understanding, robust interface management, representative testing, and proper characterization and management of SPOFs.

References

Works Cited

Evans, B. 2003. "The Galileo Trials." Spaceflight Now. Available: http://www.spaceflightnow.com/galileo/030921galileohistory.html.

GENESIS. 2005. "GENESIS Mishap Investigation Board Report Volume I." Washington, DC, USA: National Aeronautics and Space Administration (NASA).

JPL. 2000. "Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions." Special Review Board. Pasadena, CA, USA: NASA Jet Propulsion Laboratory (JPL).

ESA. 2005. "Modest Hero Sparks Team Response." European Space Agency. Available: http://www.esa.int/Our_Activities/Operations/Modest_hero_sparks_team_response.

Oberg, J. 2004. "Titan Calling: How a Swedish engineer saved a once-in-a-lifetime mission to Saturn's mysterious moon." *IEEE Spectrum.* 1 October 2004, pp. 28-33.

Renzetti, D.N. 1995. "Advanced Systems Program and the Galileo Mission to Jupiter." The Evolution of Technology in the Deep Space Network: A History of the Advanced Systems Program. Available: http://deepspace.jpl.nasa.gov/technology/95_20/gll_case_study.html.

Russell, C. 2003. *The Cassini-Huygens Mission: Volume 1: Overview, Objectives and Huygens Instrumentarium.* Norwell, MA, USA: Kluwer Academic Publishers.

Taylor, J., K.-M. Cheung, and D. Seo. 2002. "Galileo Telecommunications. Article 5." *DESCANSO Design and Performance Summary Series*. Pasadena, CA, USA: NASA/Jet Propulsion Laboratory.

Vinarcik, M.J. 2014. "Airbus A380 and Boeing 787 — Contrast of Competing Architectures for Air Transportation," in *Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering*, edited by A. Gorod et al. Boca Raton, FL, USA: CRC Press. p. 687-701.

Wall, M. 2012. "NASA Shelves Ambitious Flagship Missions to Other Planets." Space News. Available: http://www.space.com/14576-nasa-planetary-science-flagship-missions.html.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

Hubble Space Telescope

This article describes a remarkable engineering feat with vast scientific benefits and implications. The topic may be of particular interest to those facing formidable Systems Engineering (glossary) challenges where one might thrive on a thoughtful blend of humility and optimism. For addition information, refer to the links provided in Section V. Lessons Learned below.

Background

The Hubble Space Telescope (HST) Case Study was developed by the United States Air Force Center for Systems Engineering (AF CSE) located at the Air Force Institute of Technology (AFIT). The AF CSE was tasked to develop case studies focusing on the application of Systems Engineering (glossary) principle (glossary) s within various aerospace program (glossary) s. The HST study (Mattice 2005) is one of four initial case studies selected by AFIT for development in support of systems engineering graduate school instruction. The cases are structured using the Friedman-Sage framework (glossary) (Friedman and Sage 2003; Friedman and Sage 2004, 84-96), which decomposes a case into contractor, government, and shared responsibilities in the following nine concept areas:

1. Requirements Definition and Management
2. Systems Architecture Development
3. System/Subsystem Design
4. Verification/Validation
5. Risk Management
6. Systems Integration and Interfaces
7. Life Cycle Support
8. Deployment and Post Deployment
9. System and Program Management

The case study provides a useful example of the rising cost (glossary) of defect correction through successive life cycle phases, demonstrating how an error (in test fixture specification) that could have been fixed for $1,000 at the design (glossary) stage, or detected and fixed with a $10 million investment in an end-to-end test of the telescope on the ground, ended up costing $1 billion to fix when the system was in service (glossary).

Purpose

The Hubble Space Telescope (HST) is an orbiting astronomical observatory operating in the spectrum from the near-infrared into the ultraviolet. Launched in 1990 and still operational, HST carries and has carried a wide variety of instruments producing imaging, spectrographic, astrometric, and photometric data through both pointed and parallel observing programs. Over 100,000 observations of more than 20,000 targets have been produced for retrieval. The telescope is well known as a marvel of science. This case study hopes to represent the facet of the HST that is a marvel of systems engineering, which, in fact, generated the scientific research and observation capabilities now appreciated worldwide.

Viewed with the clarity that only time and hindsight provide, the HST program certainly represents one of the most successful modern human endeavors on any scale of international scope and complex (glossary) ity. As a systems engineering project the HST had to respond to requirement (glossary) s from the diverse international scientific community at a time when NASA was implementing a different research-development-acquisition philosophy and process (glossary) than what was predominately being using in most major government acquisition programs. As with most other large programs, powerful influences outside the systems engineering process itself became issues that HST Systems Engineer (glossary) s in effect had to acknowledge as integral to their overall system/program/engineering management responsibility.

Challenges

The story of how this remarkable capability came to be is a story of the complicated interactions of a systems engineering process, which we like to believe we understand, with equally demanding political, budgetary, and institutional processes we often fail to understand or comprehend at the time they occur. In the final analysis, these processes are inseparable and integral to attaining program success. The challenge to modern systems engineers is to fully embrace the discipline of the systems engineering process while at the same time learning how to continue to practice it in spite of inevitable external influences and instabilities that often cannot be anticipated.

Major differences revolved around the nature and needs of a very different HST “customer” or user from most DoD systems. The HST had to respond to requirements from the diverse international scientific community instead of from DoD’s combatant commands. In addition, at the time, NASA implemented a different research-development-acquisition philosophy and process than the DoD Acquisition Management Framework described in the DoD 5000 series acquisition reforms. As with most other large programs, powerful influences outside the systems engineering process itself became issues that HST systems engineers in effect had to acknowledge as integral to their overall system/program/engineering management responsibility.

Systems Engineering Practices

During the critical systems engineering phase for the HST program (1970s concept studies thru 1990 launch) there appears to have been no NASA systems engineering master process. Rather, field center processes were operative and possibly even in competition, as centers (especially Marshall and Goddard for HST) were in keen competition for lead management roles and responsibilities. We will see the systems engineering and program management impacts of this competition as it played out for HST, with the science mission objectives and instrumentation payloads being the motivation for Goddard vs. the vehicle/payload access to space motivation of Marshall. In the final analysis, the roles of the major contractors in engineering the system with uneven NASA participation over the system life cycle had a telling effect.

Lessons Learned

Five learning principles (LPs) were derived that address the more broadly applicable areas of systems engineering knowledge. These five LPs inform the areas of the SEBoK that are most strongly related to the case study. The five areas are:

- stakeholder requirements definition (LP1);
- planning (pre-program trade studies) (LP2);
- system integration (LP3);
- life cycle model management (LP4); and
- risk management (LP5).

A synopsis of the HST Learning Principles (LPs) are as follows:

Stakeholder Requirements Definition LP1: Early and full participation by the customer/user throughout the program is essential to success. In the early stages of the HST program the mechanism for involving the customer was not well defined. The user community was initially polarized and not effectively engaged in program definition and advocacy. This eventually changed for the better, albeit driven heavily by external political and related national program initiatives. Ultimately, institutionalization of the user’s process for involvement ensured powerful representation and a fundamental stake and role in both establishing and managing program requirements. Over time, the effectiveness of “The Institute” led to equally effective user involvement in the deployment and on-orbit operations of the system as well.

Planning LP 2: The use of Pre-Program Trade Studies (e.g. “Phased Studies” or “Phased Project Planning”) to broadly explore technical concepts and alternatives is essential and provides for a healthy variety of inputs from a

variety of contractors and government (NASA) centers. These activities cover a range of feasibility, conceptual, alternative and preliminary design trades, with cost initially a minor (later a major) factor. In the case of HST, several NASA Headquarters and Center organizations funded these studies and sponsored technical workshops for HST concepts. This approach can promote healthy or unhealthy competition, especially when roles and responsibilities within and between the participating management centers have not yet been decided and competing external organizations use these studies to further both technical and political agendas. NASA Center roles and missions can also be at stake depending on political and or budgetary realities. The systems engineering challenge at this stage is to “keep it technical, stupid!”

Systems Integration LP 3: A high degree of systems integration to assemble, test, deploy, and operate the system is essential to success and must be identified as a fundamental program resource need as part of the program baseline. For HST, the early wedding of the program to the Shuttle, prior NASA and NASA contractor experience with similarly complex programs, such as Apollo, and the early requirement for manned, on-orbit servicing made it hard not to recognize this was a big systems engineering integration challenge. Nonetheless, collaboration between government engineers, contractor engineers, as well as customers, must be well defined and exercised early on to overcome inevitable integration challenges and unforeseen events.

Life Cycle Models LP 4: Life Cycle Support planning and execution must be integral from day one, including concept and design phases. The results will speak for themselves. Programs structured with real life cycle performance as a design driver will be capable of performing in-service better, and will be capable of dealing with unforeseen events (even usage in unanticipated missions). HST probably represents a benchmark for building in system sustainment (reliability, maintainability, provision for technology upgrade, built-in redundancy, etc.), while providing for human execution of functions (planned and unplanned) critical to servicing missions. With four successful service missions complete, including one initially not planned (the primary mirror repair), the benefits of design-for-sustainment, or life cycle support, throughout all phases of the program becomes quite evident. Without this design approach, it is unlikely that the unanticipated, unplanned mirror repair could even have been attempted, let alone been totally successful.

Risk Management LP 5: For complex programs, the number of stakeholders (government and contractor) demands that the program be structured to cope with high risk factors in many management and technical areas simultaneously. The HST program relied heavily on the contractors (especially Lockheed Missiles and Space Company (LMSC) and Perkin-Elmer (P-E)), each of which “owned” very significant and unique program risk areas. In the critical area of optical systems, NASA depended on LMSC as the overall integrator to manage risk in an area where P-E was clearly the technical expert. Accordingly, NASA relied on LMSC and LMSC relied on P-E with insufficient checks, oversight, and independence of the quality assurance function throughout. While most other risk areas were no doubt managed effectively, lapses here led directly to the HST’s going to orbit with the primary mirror defect undetected, in spite of substantial evidence that could have been used to prevent this.

References

Works Cited

Friedman, G.R. and A.P. Sage. 2003. Systems Engineering Concepts: Illustration Through Case Studies. January 19, 2003. Accessed in September 2011. Available at: http://www.afit.edu/cse/docs/Friedman-Sage%20Framework.pdf.

Friedman, G. and A. Sage. 2004. “Case Studies of Systems Engineering and Management in Systems Acquisition.” Systems Engineering. 7(1): 84-96.

Mattice, J. “Hubble Space Telescope Case Study." Ft. Belvoir, VA: Defense Acquisition University (DAU). Last modified May 2, 2005. Accessed on November 27, 2012. Available at https://acc.dau.mil/adl/en-US/37600/file/9105/Hubble%20Space%20Telescope%20SE%20Case%20Study%20-%20JJ%20Mattice.pdf.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Northwest Hydro System

This case study presents the Northwest Hydro System (NwHS), which is situated in the northwestern United States (the Northwest). NwHS comprises a large number of loosely coupled autonomous elements (hydroelectric dams) that operate in a complex environment of social, regulatory, and ecological contexts. It is instructive to note that the NwHS is characterized as a project that is concerned with planning, developing, and maintaining a hydroelectric system that has evolved, and continues to evolve, over time. Each of the hydroelectric dams within the NwHS is also referred to as a project, which indicates that the individual elements of the NwHS are also evolving over time (FWEE 2016).

Background

As is shown in this case study, NwHS is an adaptive and reconfigurable system that exists within a framework of policies, rules, regulations, and agreements. The NwHS is analyzed using each of the four provisioning paradigms in SEBoK Part 4: product system engineering; service system engineering; enterprise system engineering; and system of systems engineering.

NwHS encompasses the Columbia River and its tributaries. The headwaters of the Columbia are in the Rocky Mountains of British Columbia, Canada. The river flows south into the United States and then east to west; it forms the north-south border between the states of Washington and Oregon in the U.S. and empties into the Pacific Ocean near Astoria, Oregon. The Columbia River drainage basin (Columbia and tributaries) is roughly the size of France and extends across seven U.S. states and British Columbia (Col 2016).

The Columbia River has 14 hydroelectric dams (hydro dams) on its main stem. In total, there are more than 250 hydro dams in the Columbia River Basin, each of which has a generating capacity of five or more megawatts of electricity. The NwHS produces approximately one-third of all hydroelectric power generated in the United States -

more than any other North American hydroelectric system. The amount of electrical power generated by the NwHS fluctuates between 50% and 75% of all electricity used in the Northwest; other sources include coal, natural gas, nuclear, wind, and solar. Excess electrical energy generated by NwHS is sold to other electrical grids. There are more small hydro dams than large dams. Small hydro dams have a generating capacity of 100 kilowatts to 30 megawatts; large hydro dams have a capacity of more than 30 megawatts. In addition, there are numerous micro dams that each can generate fewer than 100 kilowatts. Most micro dams provide power to isolated homes or small communities but some are elements of the NwHS and sell power to utilities (DOE 2016). Utility companies own some of the dams, some are locally and privately owned, and some are owned and operated by federal agencies.

The Bonneville Power Administration (BPA) provides about one-third of the electrical power generated by NwHS and used in the Northwest; BPA is a federal nonprofit agency – it is part of the U.S. Department of Energy. It is self-funded and covers its costs by selling electrical power generated by 31 federal hydro dams in the Columbia River Basin. The U.S. Army Corps of Engineers and the Bureau of Reclamation operate the dams.

The Bureau of Reclamation also operates a network of automated hydrologic and meteorologic monitoring stations located throughout the Northwest. This network and its associated communications and computer systems are collectively called Hydromet. Remote data collection platforms transmit water and environmental data via radio and satellite to provide near-real-time water management capabilities. Other information, as available, is integrated with Hydromet data to provide timely water supply status for river and reservoir operations (USBR 2016).

Distinguishing characteristics of individual hydro dams are their capacity to generate electrical energy and their physical structure. Several factors contribute to the differences in generating capacity of hydro dams: the number of turbines/generators; the flow rate of a river or tributary; the amount of elevation that water falls in order to spin the turbines; and environmental factors such as providing for fish passage and regulating water flow to provide irrigation water and maintain downstream ecosystems.

Dam structures include storage dams, run-of-river dams, and pumped storage dams (DOE 2016). A storage dam retains water in a reservoir for future use. A run-of-river dam harvests the energy in a river as it flows through the dam but does not impede the flow. Pumped storage dams use generated electricity to pump water back up to a storage reservoir during times of low demand for use during times of high demand. Pumped storage dams are not common in the NwHS.

Purpose

NwHS has three primary goals (NwHS 2016):

1. To provide most of the Northwest's "firm-energy" needs.
2. To maximize "non-firm" energy production.
3. To maintain the ecological environment.

Firm energy is the amount of electricity the Northwest will need each year. Planners rely on NwHS and other sources to produce enough firm energy to ensure that sufficient electricity will be generated to meet estimated needs (energy sources include hydroelectric, nuclear, coal, natural gas, solar, and wind). NwHS "firm energy" is the amount of electricity than can be generated by NwHS when the amount of available water is at a historical low, thus guaranteeing the amount of energy the NwHS can provide.

Non-firm energy is the electricity generated when the annual hydrologic cycle makes more water available for power generation than in a historically low-water year. Non-firm electricity generated by hydro dams is generally sold at a lower price than the alternatives of electricity generated by nuclear, coal, or natural gas thus making it more attractive to customers. Excess non-firm electricity is also sold to interconnected regional grids when the demand on those grids exceeds supply.

Other goals for NwHS include flood control, navigation, irrigation, and maintaining the water levels of all reservoirs.

Challenges

The NwHS is a large, complex system that has many challenges to be met.

NwHS hydro dams have varying design details (e.g., the types of turbines, generators, control systems, and fish passage facilities used). This makes routine maintenance, retrofitting, and other sustainment issues unique for each dam.

Safety and security (both physical and cyber) are common challenges for all dams; cyber security is a growing concern. Smaller dams, having fewer resources, may be more susceptible to cyber attacks than larger ones. It has been reported that on 12 occasions in the last decade hackers gained top-level access to key power networks (HLS 2010) (Cyber 2015).

Run-of-river hydro dams do not store water. They may not be able to meet their firm energy commitments when rivers are lower than anticipated and flowing slowly.

The ways in which electricity generated by a dam is transmitted and sold to utilities and large industrial customers varies widely. For instance, the Bonneville Power Administration transmits and sells power generated by federal dams. Non-federal operators must manage transmission and sale of power produced by their dams.

Depending on factors such as size, structure, location, and ownership of each dam, a large number of policies, regulations, and agreements have dramatically different effects on how dams are operated.

Some of the most contentious environmental issues are associated with maintaining the ecology of the rivers, preserving salmon and other fish, and providing sufficient irrigation water while preserving sufficient reservoir water to meet firm-energy demands (Speakout 2016).

Preserving the salmon population endangered by dams is a continuing challenge. Salmon populations have been depleted because dams impede the return of salmon to upstream spawning beds. Native Americans advocate for their traditional fishing rights, which conflict with governmental policies intended to maintain healthy salmon populations in the Columbia River Basin (Impact 2016). The National Marine Fisheries Service has recently declared that salmon recovery is a higher priority than all other purposes except flood control at 14 federal dams.

Systems Engineering Practices

The Northwest Hydro System is a large, complex system composed of loosely coupled autonomous elements; each dam operates semi-independently within a large network of similar entities and contextual constraints. The NwHS evolves over time: some dams have been retrofitted to increase power generation capability or to reconfigure connections to electrical transmission lines; new dams have been constructed; and some existing dams have been decommissioned and removed.

Human elements of NwHS include: operators; maintainers; regulators; and inspectors. Others are suppliers to NwHS (vendors and contractors); some humans are users of the electricity generated by NwHS (businesses and home owners); and some are stakeholders who depend on and are impacted by the NwHS (utilities, large industries, farmers, ranchers, homeowners, ecology advocates, Native Americans, towns).

The context of NwHS includes natural elements (rivers, terrain, weather systems, fish); elements purposefully built by humans (transmission lines, electrical grids); cyber connections (both wired and Internet); and rules, regulations, and agreements at the federal, regional, state, and local levels.

Given the complexity of the NwHS and its context, it is instructive to analyze the NwHS by applying each of the four application paradigms of systems engineering presented in SEBok Part 4: the product, service, enterprise, and system-of-systems application paradigms.

Product system provisioning

Product system provisioning applies systems engineering processes, methods, tools, and techniques to conceive, develop, and sustain the purposefully developed elements of a system (e.g., a hydro dam or a hydro system). In addition, some of the naturally occurring physical elements of a system may be shaped and configured (e.g., a river channel).

Major product elements of a hydro dam include the physical structure of the dam (including the spillway), the penstock (used to direct water into the turbines), the generating plant (i.e., the turbines used to turn generator rotors, generator stators and rotors that generate the electricity, step-up transformers used to increase the voltage level of electricity produced by the generators, and connections to transmission lines).

Cyber elements sense, measure, regulate, and control water flow, power generation, safety, security, and the structural integrity of the dam. Some turbines, for example, have adjustable vanes that are controlled to harvest maximum energy from the water, depending on the flow rate, power demand, and other factors. The cyber elements include: computing devices; supporting software (operating systems, databases, spreadsheets); data management software (collection, analysis, reporting); application software (displays of monitored status and interfaces for controlling operation of a dam); and communication interfaces to wired linkage and Internet-enabled links. In addition, software support is provided for the analog and digital devices needed to sense, measure, regulate, and control the purposefully built and naturally occurring elements of a dam and its environment.

Product system provisioning is also concerned with other issues that apply to individual dams, elements of dams, and the overall NwHS. They include issues such as: manufacturability/producibility; logistics and distribution; product quality; product disposal; conformance to policies, laws, regulations, agreements, and standards; value added for stakeholders; and meeting customer’s expectations. Many different technologies and engineering disciplines are needed to develop and sustain a hydro dam and the overall Northwest Hydro System. Product system provisioning can provide the coordination and control of systems engineering needed to develop, reconfigure, adapt, analyze, and sustain the hydro dams and the NwHS.

Service system provisioning

A service is an activity performed by an entity to help or assist one or more other entities. Service system provisioning can be applied within the various contexts of services provided by the NwHS to meet stakeholders’ requirements, users’ needs, and system interactions with operators, users, and maintainers, plus the interactions with the contextual elements that determined services provided by the NwHS in the social, business, regulatory, and physical environments.

The NWHS provides electricity to a grid that serves commercial, industrial, governmental, and domestic customers. Stakeholders in addition to customers served include those who affect or will be affected by development, operation, and sustainment of a dam. Downstream stakeholders served, for example, include, Native Americans, farmers and ranchers, and communities that receive the service of water released by the dam.

Additional service attributes include: the services that enable operators and maintainers to efficiently and effectively operate and maintain the physical and cyber elements of a dam; release water from the dam in a manner that services the upstream and downstream ecosystems; manage sharing of electrical power with other regional grids; provide emergency responses to power demands that result from electrical brownouts, blackouts, and overloads; and handle system failures that might be caused by earthquakes, terrorist attacks, and other catastrophic events.

Enterprise system provisioning

An enterprise, such as the NwHS, consists of one or more organizations that share a mission, goals, and objectives to offer an output such as a product or service. The mission and goals of the NwHS are to provide most of the Northwest's firm energy needs and to maximize non-firm energy production while serving stakeholders and preserving affected environmental ecosystems. To meet those goals, the Northwest Hydroelectric Association (NWHA) coordinates the planning, design, improvement, and operation of the hydro dams that constitute the NwHS enterprise.

NWHA members represents all segments of the hydropower industry – independent developers and energy producers; public and private utilities; manufacturers and distributors; and local, state and regional governments including water and irrigation districts. Other NWHA members include contractors, Native American tribes, and consultants: engineers, financiers, environmental scientists, attorneys and others (NWHA 2016).

Note that an enterprise may consist of multiple organizations that are engaged in a common endeavor. The NwHS is a large complex enterprise that has many constituent organizations; namely, the organizations that own and operate the hydro dams and the other stakeholder members of the NWHA. Differences in ownership, structure, location, and size of hydro dams, the special interests of various NWHA members, and a complicated regulatory process, are some of the distinguishing characteristics of the NwHS that can be analyzed by enterprise systems provisioning.

System of Systems provisioning

Many systems are composed of autonomous elements that are combined to provide increased capabilities that cannot be provided by the elements operating in isolation. The Northwest Hydro System is a system of systems comprised of autonomous hydro dams that have different owners, different operators, different stakeholders, and different regulators. The autonomous hydro dams could not provide the NwHS capabilities without the overall coordination and control that can be managed by applying system of systems provisioning.

Lessons Learned

NwHS is a collection of many interrelated ongoing projects that have shared common goals and shared constraints. The unique characteristics of NwHS make it a useful case study to illustrate how the four provisioning paradigms in SEBoK Part 4 provide essential viewpoints for analyzing large complex systems comprised of loosely coupled, autonomous elements.

Product systems engineering allows the collection of physical and purposefully built NwHS elements and their interconnections to be analyzed by applying systems product engineering processes and methods.

Service systems engineering supports analysis of the NwHS services provided to customers, users, farmers, ranchers, Native Americans, and other stakeholders who rely on NwHS for those services.

Enterprise systems engineering considers the broad scope and impact of the NwHS enterprise, both positive and negative on the northwestern United States within the context of economic, social, physical, and regulatory environments.

System of Systems engineering applies the principles of planning, coordination, and operation to a collection of semi-autonomous hydro dams that form the Northwest Hydro System. The complexity of adding new dams as well as modifying and decommissioning existing dams in a seamless manner can best be understood by applying system of systems engineering processes and methods.

Taken together, the four provisioning paradigms in SEBoK Part 4 present a comprehensive view of a very large complex system whose many dimensions would be otherwise difficult, if not impossible, to comprehend when the NwHS is examined using only one of the paradigms.

References

Works Cited

Col. 2016. Columbia River. Available at https://en.wikipedia.org/wiki/Columbia_River Accessed February 15, 2016..

Cyber. 2015. US in Fear of New Cyber Attack. Available at http://www.independent.co.uk/news/world/americas/bowman-avenue-dam-us-in-fear-of-new-cyber-attack-as-dam-breach-by-iranian-hackers-is-revealed-a6782081.html Accessed February 16, 2016.

DOE. 2016. Types of Hydropower Plants. Accessed February 16, 2016. Available. http://energy.gov/eere/water/types-hydropower-plants.FWEE. 2016. Foundation for Water and Energy Education. Accessed February 17, 2016. Available: http://fwee.org/about-fwee/about/HLS. 2010. Dam Sector Roadmap to Secure Systems. Available at http://www.damsafety.org/media/Documents2/security/files/DamsSectorRoadmapToSecureControlSystems.pdf Accessed February 16, 2016.

Impact. 2016. The impact of the Bonneville Dam on Native American Culture. Available at http://www2.kenyon.edu/projects/Dams/bsc02yogg.html Accessed February 16, 2016.

NWHA. 2016. Northwest Hydroelectric Association. Available at http://www.nwhydro.org/nwha/about/ Accessed February 17, 2016.

NwHS. 2016. How the Northwest Hydro System Works. Available: http://fwee.org/nw-hydro-tours/how-the-northwest-hydro-system-works/Accessed February 15, 2016.

USBR. 2016. Hydromet. Available at http://www.usbr.gov/pn/hydromet/Accessed February 16, 2016.

Speakout. 2016. Stop the Hydropower Wish List Bill. Available: http://act.americanrivers.org/page/speakout/stop-unlockhydro-senate?source=adwords& gclid=CPTF7u3G_MoCFVE0aQodiGgMQg Accessed February 16, 2016.

Primary References

FCRPS. 2003. Federal Columbia River Power System. Available at https://www.bpa.gov/power/pg/fcrps_brochure_17x11.pdf Accessed February 16, 2016.

HLS. 2010. Dam Sector Roadmap to Secure Systems. Available at http://www.damsafety.org/media/Documents2/security/files/DamsSectorRoadmapToSecureControlSystems.pdf Accessed February 16, 2016.

NWHA. 2016. Northwest Hydroelectric Association. Available: http://www.nwhydro.org/nwha/about/Accessed February 17, 2016.

NwHS. 2016. How the Northwest Hydro System Works. Available: http://fwee.org/nw-hydro-tours/how-the-northwest-hydro-system-works/.Accessed February 15, 2016.

Additional References

FWEE1. 2016. Foundation for Water and Energy Education. Available: http://www.fwee.org. Accessed February 16, 2016.

FWEE2. 2016. Overview of Hydropower in the Northwest. Available: http:/ / fwee. org/ education/ the-nature-of-water-power/overview-of-hydropower-in-the-northwest/Accessed February 16, 2016.

HP. 2016. Hydro Portal. Available at https://energypedia.info/wiki/Portal:Hydro Accessed February 16, 2016.

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

Singapore Water Management

This example was produced as a SE example directly for the SEBoK. It describes a systems engineering approach in the development of a sustainable National Water Management System for the Republic of Singapore. It demonstrates the successful outcome of long term planning and a systems approach to preempt a critical water shortage. The example is primarily based on information taken from a paper presented at the INCOSE International Symposium in 2008. (Chia 2008)

Description

When Singapore achieved independence in 1965, water supply depended on water catchment in local reservoirs and two bilateral water agreements with its closest neighbor, Malaysia. These water agreements are registered with the United Nations. The first agreement expired in August 2011, and the second agreement will expire in 2061 (Singapore 2012). After several failed attempts to renegotiate the extension of the first water agreement, Singapore determined that it was necessary to achieve full water self sufficiency by 2060 in case the second water agreement also could not be extended. An intermediate goal was to match the supply of the first water agreement before it expired. This was achieved in several ways. In 2001, the Public Utilities Board (PUB), the national water agency responsible for treating raw water in Singapore, was charged to also begin managing wastewater and stormwater, allowing for an integrated and holistic approach to water management.

This example examines Singapore's water management system from a large-scale systems engineering perspective, particularly focusing on the goals, boundaries (see Concepts of Systems Thinking), stakeholders (see Stakeholder Needs and Requirements), and complexities involved in this type of a national system. This approach illustrates how Systems Thinking (illustrated through causal loop diagrams) and other systems engineering tools may be used to understand systems complexities. Concepts and methodologies of learning organizations were applied to enable understanding of behavioral complexities. Lean thinking facilitated a long term strategic philosophy, built on the premise of continuous improvements.

Perhaps more importantly, it shows that while systems engineering, especially the Systems Approach, is necessary for the conceptualization and planning of such a complex system, it is not sufficient for success. It is the systemic structures that have been put in place over decades, political will, leadership, people, and culture that make such tasks realizable.

The supply of water in Singapore is managed in totality. Collecting rainwater, purchasing water, purifying water utilizing reverse osmosis and desalination were all considered. Approaches included even incentivising consumers to change their habits by making drains and canals recreational areas to encourage the public not to dispose of waste in their drains. By managing sewage and drainage together with water, environmental considerations are taken into account as well. By carefully adjusting organizational boundaries, Singapore has managed to reduce silo thinking and parochial interests. The relationships between the industry innovators, government, suppliers and users, and

technology innovators create opportunities for Singapore's water management. This demonstrates how multiple stakeholder interests can be combined to create a viable water management solution. Continuous improvements through the use of technology and elimination of waste, such as reducing water that is not accounted for in the system, help to assure the sustainability of an adequate supply of water for a growing Singapore population. The Singapore Water Management system is already in successful operation and is being studied by the Organisation for Economic Co-operation and Development (OECD) and by other nations.

Summary

The supply of water in Singapore is managed in totality through a systems approach, i.e., water catchment, supply, sewage and drainage. The importance of relationships between the stakeholders is also recognized. Industry innovators, political leadership, suppliers, and consumers are all involved; the project has been able to incentivize this diverse group to work together for a common goal, i.e., assuring the sustainability of an adequate supply of water for Singapore into the future.

Utilizing systems engineering and taking into consideration the systemic structures and culture required has helped Singapore achieve its first milestone of supplying its own water resources by 2010. Singapore has been able to overcome the shortfall that would have come about with the expiry of the first water agreement with Malaysia in 2011.

References

Works Cited

Chia, A. 2008. "A Large-Scale Systems Engineering Perspective of Water Management in Singapore." Proceedings of the 18th Annual INCOSE International Symposium, 15-19 June 2008, Utrech, The Netherlands.

Singapore Government. 2012. "The Singapore Water Story." Accessed August 2012. Available at http://www.pub.gov.sg/water/Pages/singaporewaterstory.aspx.

Primary References

None.

Additional References

Public Utilities Board. 2007. "PUB Main Website". Accessed August 2011. Available at http://www.pub.gov.sg.

Tortajada, C. 2006. "Water Management in Singapore." *International Journal of Water Resources Development.* 22(2): 227-240.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Submarine Warfare Federated Tactical Systems

This article describes the transformation of the systems engineering and integration program that produces the common combat system used across the United States Navy (USN) submarine fleet from traditional document-based systems engineering (DBSE) to model-based systems engineering (MBSE). The topic may be of particular interest to those dealing with programs in the sustainment and evolution phase of their life cycle. For additional information, refer to the links provided in Section V, Lessons Learned below.

Background

Modern submarines are typically in service for 20 - 40 years. Submarines and their internal systems are typically state-of-the-practice at launch, but most navies find it necessary to upgrade the ship's combat system at least once during the operational lifetime. The evolution of threats, technology and interoperability drives the USN to upgrade their submarine combat systems and key components including the sonar (Fages 1998) (Ford and Dillard 2009) and tactical control systems continuously (Jacobus and Barrett 2002).

Over the last three decades submarine combat systems have evolved from multiple independent systems (sonar, combat control, imaging, electronic warfare, weapon control, etc.) with manual or point-to-point interfaces into networked federations of systems (FoS). Confusingly, these component systems are often referred to as subsystems in the literature. Typically, each new class of submarines has been equipped with a new combat system, with a corresponding logistics and sustainment tail unique to that class.

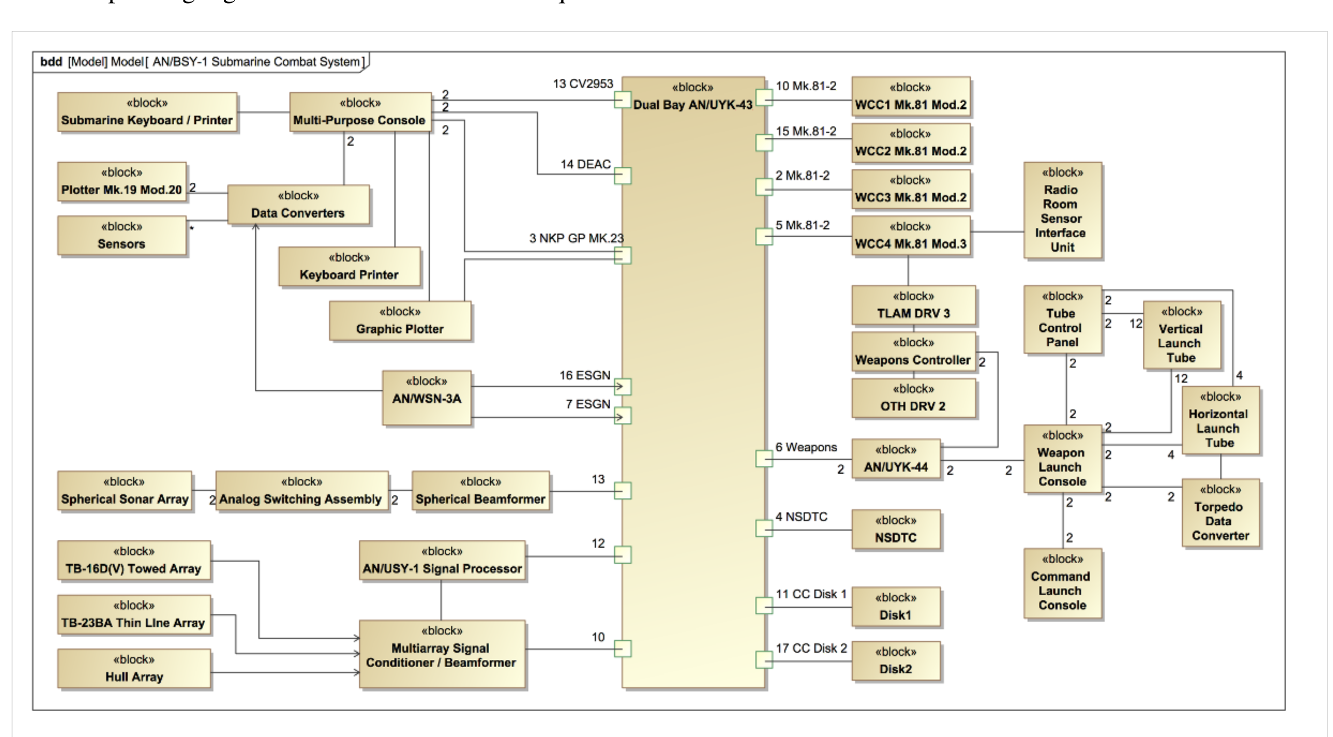


Figure 1. SysML Block Definition Diagram of 1985 era submarine combat system composed from independent component systems with point-to-point interfaces (SEBoK Original)

In the USN, each of these component systems has its own acquisition program, customer, and contractor team. Starting as legacy military systems hosted on traditional military-unique computational platforms, these systems have evolved to utilize Commercial Off-The-Shelf (COTS) computational and networking platforms, and leverage large amounts of COTS software.

As the component systems became more tightly interconnected, the acquisition customers established and collaboratively funded a systems engineering and integration (SE&I) program to manage the interfaces between

systems, manage technology insertion and the obsolescence of common COTS hardware, and to integrate and test the production systems (Cooper, Sienkiewicz and Oliver 2006). Starting with the Virginia class, this SE&I program was expanded to encompass both new-production and in-service submarine modernization efforts. Over time, the combat systems of the various submarine classes were converged into variants of a single product line (Zingarelli, et al. 2010). In addition, the independent system-to-system interfaces were rationalized where practical into common interfaces shared between multiple systems.

Purpose

The submarine combat system SE&I program delivers an updated production baseline annually, along with product line variants for each submarine class or subclass being built or upgraded that year. Production systems implementing this baseline are delivered to new-build submarines, and to in-service submarines being upgraded on a roughly six-year cycle. The common combat system product line is referred to as the Submarine Warfare Federated Tactical Systems (SWFTS). SWFTS is deployed by the USN on submarines of the Los Angeles (SSN 688), Ohio (SSGN 726, SSBN 730), Seawolf (SSN 21), and Virginia (SSN 774) classes, and by the Royal Australian Navy on the Collins (SSG 73) class. SWFTS is also planned for the next-generation USN Columbia (SSBN) and RAN Future Submarine classes. Compared to the submarine combat systems that it replaced, SWFTS significantly reduces development, maintenance and training costs while delivering enhanced combat capabilities and facilitating the rapid insertion of new or improved capabilities (Zingarelli, et al. 2010).

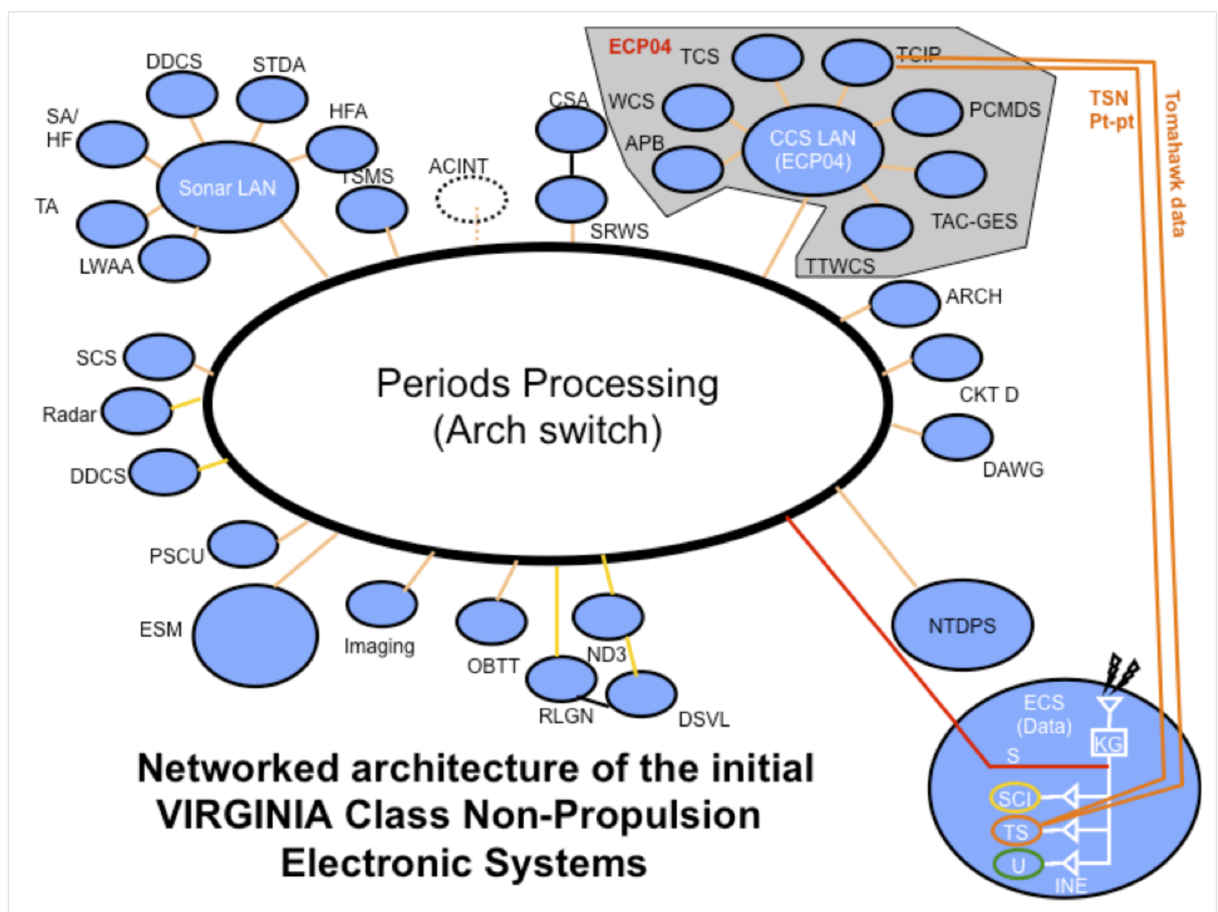


Figure 2. Contemporary submarine combat systems are networked with converged data interfaces (Produced by Lockheed Martin for US Navy. Approved for Public Release by US Navy, #16-348, June 2016)

Challenges

The USN submarine fleet encompasses substantial platform variability between class, sub-classes, and even individual ships within a sub-class. The RAN Collins class contributes additional variability. Platform variability drives combat system variability.

SWFTS is a Federation of Systems (FoS), with each platform hosting a subset of 40 systems produced by 20 different program offices. As is common with System of Systems (SoS) and FoS, there is no central program office that can command the compliance of all of the component system programs. Instead, the evolution of SWFTS is executed through negotiation and consensus.

Many baselines must be produced each year: new common hardware baselines are introduced in odd years, while new common software baselines are introduced in even years (Jacobus, Yan and Barrett 2002). In addition, multiple incremental developmental baselines are established each year. Once the annual production baseline for the product line is defined, variants must be developed for each submarine class or subclass built or upgraded that year (Mitchell 2012).

Like most other defense programs, the SWFTS SE&I program is under constant pressure to accomplish more with decreasing resources. There has been steady increase in SE scope despite decreasing budgets. Program leadership has responded in part through continuous SE process improvement. Improvements have included test automation, changes in the requirements management processes and tools (spreadsheets to IBM® Rational® DOORS® to OMG® SysML®), refined tooling for change management, and the DBSE to MBSE transition that is the focus of this case study. Substantial Return on Investment (ROI) has been achieved with each major SE process or tooling improvement (Mitchell 2014).

Systems Engineering Practices

During 2009 the SWFTS SE&I program conducted a Model Driven Architecture (MDA) study to determine if MDA should be the next step in the program's continuous SE process improvement. The MDA study predicted a positive Return on Investment (ROI) from converting to MBSE. In January 2010 the SWFTS SE&I program office initiated a three-year effort to develop and validate a SWFTS FoS model and MBSE process. In 2013 a SWFTS baseline was developed using both DBSE and MBSE in parallel. The DBSE products were used to validate the MBSE products. Based on that successful validation, the SWFTS SE&I program transitioned to MBSE for all ongoing work.

Until the transition in 2013, SWFTS SE was performed using traditional DBSE. Requirements were managed in DOORS, and reviewed and used by engineers in the form of massive spreadsheets with hundreds of columns and thousands of rows. The design of each variant was documented in Microsoft Office files. Baseline change requests (BCR) were documented in briefings, and analyzed by all component system programs in parallel for potential impact. Approved BCRs were manually merged into DOORS and into revised baseline documents for each effected variant.

Starting in 2010, the customer community invested in a three-year MBSE transformation effort. The engineering team performed an in-depth tool trade study to select and set up the MBSE environment. That trade study resulted in the selection of MagicDraw™ for system modeling, using Teamwork™ as the model repository.

Once the modeling environment was installed, the MBSE transformation team architected, developed and populated a SysML-based model of the SWFTS FoS interfaces. The team updated the SWFTS SE process to take advantage of the new MBSE environment, with the constraint that the MBSE process produce SE products that were effectively identical to those produced by the DBSE process.

In 2013 the new MBSE environment and process was used to produce a set of SWFTS baseline SE products. In parallel, the DBSE process produced equivalent products. The two sets of baseline products were compared in detail, with all differences traced back to root causes.

This analysis identified a significant number of minor differences that were all traced to errors in the DBSE products. This validated the MBSE process, and demonstrated that while those initial MBSE baseline products were more labor-intensive than the DBSE baseline products, the new MBSE process produced higher quality products. After this validation, the SWFTS SE&I program switched over to MBSE as their basic process.

Since that transformation, SE process improvement has continued apace. Requirements management has moved from DOORS to the system model in MagicDraw. As of 2016, the system model is the baseline for requirements, architecture and the FoS design. BCR impact analysis is now performed in model, leveraging capabilities of the toolset for automated assistance. Variants are documented in the system model as system configurations. Most SE products are generated from system model on demand.

While the initial scope of MBSE was limited to managing the interfaces between component systems, once the transition was successful MBSE started expanding out to encompass additional SWFTS SE&I tasks. MBSE is now beginning to spread into the component system programs as well as the overall submarine combat system SE&I program.

Lessons Learned

Seven learning principles (LPs) (Friedman and Sage 2005) were derived that address the more broadly applicable areas of systems engineering knowledge, and inform the areas of the SEBoK that are most strongly related to the case study. They are:

- Requirements traceability (LP1);
- Communications (LP2);
- Productivity (LP3);
- Quality (LP4);
- Managing Change (LP5);
- Managing variants (LP6); and
- Life cycle (LP7).

Requirements traceability LP1: MBSE improves traceability in multiple dimensions, but maintaining requirements traceability between a traditional database and the MBSE system model can be challenging. While DOORS, MagicDraw and Teamwork can interoperate to provide requirements management and traceability, the combination is fragile. Without careful configuration management, synchronization can lead to database corruption. If DOORS and Teamwork are on separate servers, maintaining the connection can run afoul of ever-evolving corporate information assurance (IA) policies.

Requirements can be managed using the SysML language inside the system model quite effectively. This approach can reduce the resources needed to keep the system model in sync with a traditional requirements database system and increase overall SE productivity.

Communications LP2: Tailored SE products generated from the system model can substantially enhance communications both within the technical team and between customer stakeholders. Graphical depictions of the system model often communicate better to human stakeholders than massive spreadsheets and textual documents. Further, the enhanced precision driven by modeling can reduce miscommunications between both technical and programmatic stakeholders.

Having the architecture and design in a system model makes it affordable to generate specialized SE products on demand for particular communications needs while keeping all SE products in concordance. Even technical stakeholders who thought they understood the design can find new insights by looking at it in different representations.

Productivity LP3: MBSE increases productivity by enhancing communications within the team, automation of routine tasks and through cost avoidance. Processes used in DBSE often require substantial revision to achieve the

potential productivity gains of MBSE. In particular, review processes must be modified to take advantage of the tooling.

The selected modeling tools constrain how you can practically re-engineer SE processes. Automation can replace a great deal of routine SE work (document generation, identifying potential impacts of changes, etc.).

Developing strong modeling style guidelines and specialized representations, along with training materials to indoctrinate new team members as they join the program, is worth the investment. MBSE does require a trained cadre of modelers, but not all systems engineers have to become skilled modelers.

To effectively quantify the benefits of MBSE, a program needs to plan metrics collection carefully, and then stick to the plan long enough to collect meaningful data.

Quality LP4: Much of the ROI from the MBSE transition can be in improved quality. Improving the quality of SE products reduces latent defects in systems delivered to the customer, reducing maintenance costs and increasing customer satisfaction

Models are less tolerant of imprecision than documents. The increased precision improves SE product quality, both in reduced defect generation and in reduced defect escape. The automation of product generation can make specialized SE products affordable, further enhancing system quality.

Managing change LP5: How a proposed system change is understood and executed is fundamentally different between a model- and a document-centric approach. In the document-centric approach, the focus is on “What should my final output look like?” In the model-based approach, the focus is on “What does this change mean to the system? Which other parts of the system are impacted by this change?”

Change management is hard. When moving from DBSE to MBSE you need to think carefully up front about what approach you are going to take, and then design the system model to facilitate that approach. Change management also impacts tool selection, since different tools align better with different approaches.

Managing variants LP6: The most common process for managing variants in DBSE is ‘clone and own’, where each new product family member takes the then-current baseline and ‘forks’ the baseline for evolution of the variant. This makes synchronizing changes to the core baseline across the product family a very labor-intensive process. Treating variants as deviations from a core baseline in a model greatly reduces the cost of managing variation in a product family.

Variant management is hard. You need to think about what approach you plan to take up front, and design your system model to accommodate it. The selected approach impacts tool selection and tailoring.

Design the system model to treat variants as deviations from the core baseline. Then changes to the core baseline are automatically shared among all variants, and impact to product family members is limited to any impact of core baseline change on specific variant deviations. This also facilitates commonality between variants, a key customer goal as commonality reduces logistics and training costs.

Life cycle LP7: MBSE can be applied early or late in the product family life cycle. While most projects using MBSE start off model-based, a program can transition to MBSE late in the life cycle.

Getting from DBSE to MBSE requires serious engineering, careful thought, planning and implementation. The SWFTS MBSE transition required three years of investment by the customer. That time and budget was spent primarily in designing and developing the system model and in re-engineering the SE processes.

Start the transition with carefully defined scope. Once that is accomplished you can expand the scope of MBSE from there.

References

Works Cited

Cooper, Dennis J., Joan Sienkiewicz, and Mike Oliver, "System Engineering and Integration for Submarine Combat Systems in the COTS Environment", NDIA Systems Engineering Conference, San Diego, CA, 23-26 October 2006

Fages, H I. Submarine programs: A resource sponsor's perspective.", *The Submarine Review*, (July 1998): 53–59.

Ford, David N., and John T. Dillard, "Modeling open architecture and evolutionary acquisition: implementation lessons from the ARCI program for the Rapid Capability Insertion process ", *Proceedings of the Sixth Acquisition Research Symposium: Defense Acquisition in Transition* 2 (Apr 2009): 207- 235.

Friedman, G.R. and A.P. Sage, Case studies of systems engineering and management in systems acquisition." S*ystems Engineering,* Vol. 7, No. 1, (2004) 84-97

Jacobus, P., P. Yan, and J. Barrett, "Information management: the Advanced Processor Build (Tactical)", *JOHNS HOPKINS APL TECHNICAL DIGEST*, Vol. 23, No. 4 (Jan 2002): 366-372.

Zingarelli, Mark A, Steven R. Wright, Robert J. Pallack, and Kevin C. Matto, SWFTS - System Engineering Applied to Submarine Combat Systems,*Engineering the Total Ship,* Falls Church, VA, July 2010.

Primary References

Mitchell, Steven W., Efficiently Managing Product Baseline Configurations in the Model-Based System Development of a Combat System Product Family," *INCOSE International Symposium*, Rome, Italy, July 2012

Mitchell, Steven W., "Transitioning the SWFTS Program Combat System Product Family from Traditional Document-Centric to Model-Based Systems Engineering", *Journal of Systems Engineering*, Vol. 17, No. 2, Spring 2014

Additional References

Gibson, B., S. W. Mitchell, and D. Robinson, "Bridging the Gap: Modeling Federated Combat Systems", *Third International Conference on Model Based Systems Engineering*, Fairfax, VA, Sept 2010

Mitchell, Steven W., Model-Based System Development for Managing the Evolution of a Common Submarine Combat System," A*FCEA-GMU C4I Center Symposium on Critical Issues in C4I,* 18-19 May 2010

Mitchell, Steven W., Complex Product Family Modeling for Common Submarine Combat System MBSE," Th*ird International Conference on Model Based Systems Engineering,* Fairfax, VA, Sept 2010.

Mitchell, Steven W., Efficient Management of Configurations in the Model-Based System Development of a Common Submarine Combat System," A*FCEA-GMU C4I Center Symposium on Critical Issues in C4I,* 24-25 May 2011

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

UK West Coast Route Modernisation Project

This example was created as a SE example directly for the SEBoK. it describes the United Kingdom West Coast Main Line railway project and some of the problems which occurred on this project before the implementation of SE. It also discussed the value of applying some aspects of SE, even if this is done later in the project.

This example is based on information from a report by the UK National Audit Office (NAO 2006). it also uses an INCOSE publication on systems engineering case studies (INCOSE 2011) to help structure its conclusions.

Description

The West Coast Main Line (WCML) is a principal United Kingdom (UK) railway artery serving London, the Midlands, the North West and Scotland. The Line is responsible for over 2,000 train movements each day, with more than 75 million rail journeys made each year on the route. It accounts for 43% of Britain's UK freight market (Railway People 2011). In 1998, the British government embarked on a modernization program called the West Coast Route Modernisation (WCRM) project, to carry out a significant volume of modernization work between 1998 and 2008, delivering increased capacity and reduced journey times as well as replacing worn-out parts of the railway. It was challenging a job involving 640 kilometers of track—much of which was incapable of carrying high-speed rail cars. Some sections were seriously dilapidated, and new trains would require a complete overhaul of signaling, power supply, and switching systems.

Early on, the WCRM upgrade had serious problems. A major complicating factor was the introduction of a new signaling technology that was designed to allow improved services for new trains running at 140 miles per hour. By 2001, neither the rail infrastructure upgrade nor the new trains were on course for delivery as expected in the 1998 agreement. By May 2002 the projection of the program's final cost had risen from £2.5 billion (in 1998) to £14.5 billion, but had delivered only a sixth of the original scope.

In January 2002, the UK Secretary of State instructed the Strategic Rail Authority (SRA) to intervene and find a way to renew and upgrade the WCML. An SRA analysis identified the following issues:

- The program lacked direction and leadership before 2002.
- The project did not have a delivery strategy and there was no central point for responsibility and communication.
- There was a lack of openness and communication regarding the program with interested parties before 2002 and a lack of stakeholder management.
- Scope changes arose because WCRM did not have an agreed-upon specification that matched required outputs with inputs.
- There was inadequate knowledge about the West Coast asset condition.
- Technology issues related to the decision to replace conventional signaling with unproven moving block signaling introduced major risk into deliverability and cost before 2002. These technology issues caused scope changes and program delay.
- Project management (PM) was weak, with a lack of senior management skills, too many changes in personnel, and ill-defined and fragmented roles and responsibilities. There was no integrated delivery plan and there was limited oversight of contractors. Poor management of contracts added to costs.

In order to remedy the situation the SRA initiated the following actions, which align with generally accepted systems engineering (SE) practice:

- A clear direction for the project was developed and documented in the June 2003 West Coast Main Line Strategy, specifying desired goals and outcomes.
- A clear, measurable set of program outputs was established, along with more detailed infrastructure requirements, which were then subject to systematic change control and monitoring procedures fixing scope. Contractors were invited to tender complete detailed designs and deliver the work to a fixed price.

- Clear program governance structures were instituted.
- The SRA consulted widely with stakeholders and, in turn, kept stakeholders informed.

A National Audit Office (NAO) report concluded that the new arrangements worked well and that there were benefits to this approach. (NAO 2006) Until this time, one of the program's key constraints and cost drivers had been the ability to access certain areas of the track. The new approach facilitated the ability to obtain possession of the track for engineering work, which was crucial to delivery. The new approach also enabled the program to identify opportunities to reduce the total cost by over £4 billion.

The NAO report also discussed a business case analysis by the SRA that identified the following benefits (NAO 2006):

- benefit:cost ratio for the enhancements element was 2.5:1;
- journey times and train frequencies exceeded the targets set out in the 2003 West Coast Strategy;
- growth in passenger numbers exceeded expectations (e.g., by 2005-06, following Phase 1 of the West Coast program, annual passenger journeys on Virgin West Coast grew by more than 20%); and
- punctuality improved (e.g., by September 2006, average time delays on Virgin West Coast trains have been approximately 9.5 minutes, a 43% improvement on the average delay of 17 minutes in September 2004).

The WCRM problems could be associated with a number of systems engineering concepts and practices: stakeholders requirements, planning, analysis of risks and challenges of new technology and associated risk management, decision management, configuration or change management, information management, and management oversight.

Summary

The WCRM project illustrates that when SE concepts and practices are not used or applied properly, system development can experience debilitating problems. This project also demonstrates how such problems can be abated and reversed when SE principles and methods are applied.

References

Works Cited

INCOSE Transportation Working Group. 2011. *Systems Engineering in Transportation Projects: A Library of Case Studies*, version 1.0. Seattle, WA, USA: International Council on Systems Engineering. March 9th, 2011.

NAO. 2006. *The Modernisation of the West Coast Main Line, Report by the Comptroller and Auditor General*. London, UK: National Audit Office. November 22, 2006. HC 22 Session 2006-2007.

Railway People. 2011. '"West Coast Route Modernisation." RailwayPeople.com website. Accessed July 25, 2011. Available at: http://www.railwaypeople.com/rail-projects/west-coast-route-modernisation-3.html.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

Virginia Class Submarine

This example was developed as a SE example directly for the SEBoK. it describes the Virginia Class submarine sonar system project. In particular the approach taken to the development of a sonar system architecture and how this helped in the integration of commercial off the shelf products.

Description

Prior to the Virginia class submarine, sonar systems were comprised of proprietary components and interfaces. However, in the mid-1990s the United States government transitioned to the use of commercially developed products - or commercial off the shelf (COTS) products - as a cost-saving measure to reduce the escalating costs associated with proprietary-based research and development. The Virginia class submarine system design represented a transition to COTS-based parts and initiated a global change in architectural approaches adopted by the sonar community. The lead ship of the program, Virginia, reduced the number of historically procured parts for nuclear submarines by 60% with the use of standardization. The Virginia class submarine sonar system architecture has improved modularity, commonality, standardization, and reliability, maintainability and testability (RMT) over historical sonar systems.

Architectural Approach: Standardization

Based on the new architectural approach and the success of the transition, system architecture experts developed an initial set of architecture evaluation metrics:

- Commonality
  - Physical commonality (within the system)
    - Hardware (HW) commonality (e.g., the number of unique line replaceable units, fasteners, cables, and unique standards implemented)
    - Software (SW) commonality (e.g., the number of unique SW packages implemented, languages, compilers, average SW instantiations, and unique standards implemented)
  - Physical familiarity (with other systems)
    - Percentage of vendors and subcontractors known
    - Percentage of HW and SW technology known
  - Operational commonality
    - Percentage of operational functions which are automated
    - Number of unique skill codes required
    - Estimated operational training time (e.g., initial and refresh from previous system)
    - Estimated maintenance training time (e.g., initial and refresh from previous system)
- Modularity
  - Physical modularity (e.g., ease of system element or operating system upgrade)
  - Functional modularity (e.g., ease of adding new functionality or upgrading existing functionality)
  - Orthogonality
    - Level to which functional requirements are fragmented across multiple processing elements and interfaces
    - Level to which throughput requirements span across interfaces
    - Level to which common specifications are identified
  - Abstraction (i.e., the level to which the system architecture provides an option for information hiding)
  - Interfaces
    - Number of unique interfaces per system element

        - Number of different networking protocols
        - Explicit versus implicit interfaces
        - Level to which the architecture includes implicit interfaces
        - Number of cables in the system
- Standards-based openness
    - Interface standards
        - Ratio of the number of interface standards to the number of interfaces
        - Number of vendors for products based on standards
        - Number of business domains that apply/use the standard (e.g., aerospace, medical, and telecommunications)
        - Standard maturity
    - Hardware standards
        - Ratio of the number of form factors to the number of line replaceable units (LRUs)
        - Number of vendors for products based on standards
        - Standard maturity
    - Software standards
        - Number of proprietary and unique operating systems
        - Number of non-standard databases
        - Number of proprietary middle-ware
        - Number of non-standard languages
    - Consistency orientation
        - Common guidelines for implementing diagnostics and performance monitor/fault location (PM/FL)
        - Common guidelines for implementing human-machine interface (HMI)
- Reliability, maintainability, and testability
    - Reliability (fault tolerance)
    - Critical points of fragility (e.g., system loading comprised of percent of processor, memory, and network loading)
    - Maintainability (e.g., expected mean time to repair (MTTR), maximum fault group size, whether the system can be operational during maintenance)
    - Accessibility (e.g., space restrictions, special tool requirements, special skill requirements)
    - Testability
        - Number of LRUs covered by built-in tests (BIT) (BIT coverage)
        - Reproducibility of errors
        - Logging/recording capability
        - Whether the system state at time of system failure can be recreated
        - Online testing (e.g., whether the system is operational during external testing and the ease of access to external test points)
        - Automated input/stimulation insertion

Other Points

The Virginia class submarine acquisition exhibited other best practices. These are discussed by Schank (2011), GAO (2008), and General Dynamics (2002).

These best practices included stringent design trades to keep costs under control, careful consideration of technical maturity of components, and the importance of program stability.

Summary

In summary, the work on the Virginia class submarine prompted a change in the traditional architectural approach used in the sonar community to design submarine sonar and validated the cost savings in both research and development (R&D) and in component costs when transitioning from proprietary interfaces to industry standard interfaces. The identification of a list of feasible architecture evaluation metrics was an added benefit of the effort.

References

Works Cited

GAO. 2008. *Defense Acquisitions: Assessment of Selected Weapon Programs Report*. Washington, DC, USA: US. Government Accountability Office (GAO). March 2009. GAO-09-326SP.

GD Electric Boat Division. 2002. *The Virginia Class Submarine Program: A Case Study*. Groton, CT: General Dynamics. February, 2002.

Schank, J.F. et al. 2011. *Learning from Experience, Volume 2: Lessons from the U.S. Navy's Ohio, Seawolf, and Virginia Submarine Programs.* Santa Monica, CA, USA: Rand. Available at http://www.rand.org/content/dam/rand/pubs/monographs/2011/RAND_MG1128.2.pdf [1]

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

References

[1] http://www.rand.org/content/dam/rand/pubs/monographs/2011/RAND_MG1128.2.pdf

Combined Examples

Applying a Model-Based Approach to Support Requirements Analysis on the Thirty-Meter Telescope

This article describes how a model-based systems engineering (MBSE) approach was used to support requirements analysis, system design, and early verification for critical subsystems of the Thirty Meter Telescope (TMT) [8]. The MBSE approach applied the Executable Systems Engineering Method (ESEM) [4] [5] and the Open-source Engineering Environment (OpenMBEE) [7] to specify, analyze, and verify requirements of TMT's Alignment and Phasing System (APS) and the Narrow Field Infrared Adaptive Optics System (NFIRAOS). The value proposition for applying this MBSE approach was to establish precise requirements and fine-grained traceability to system designs, and to verify key requirements using executable SysML [6] models beginning early in development.

Background

The TMT (Figure 1) is a next-generation ground-based extremely large telescope designed to answer key science questions regarding the nature and composition of the Universe. At its core is a wide-field, altitude-azimuth Ritchey-Chrétien design with a 492 segment, 30-meter diameter primary mirror (M1), a fully active secondary mirror, and an articulated tertiary mirror. Each segment's optical performance is sensitive to three rigid body degrees of freedom: piston, tip, and tilt. To obtain optimal image quality, the segmented M1 must perform like a single monolithic mirror, achieved through a multitude of controls working to co-align, co-focus, and co-phase its segments. The APS (Figure 2) is responsible for the overall pre-adaptive optics wavefront quality, using starlight to measure wavefront errors and align the TMT optics. Adaptive optics systems like the NFIRAOS (Figure 2) are designed to sense real-time atmospheric turbulence and correct the telescope's optical beam to remove its effect, enabling diffraction-limited imaging on the ground. In LGS MCAO and NGSAO modes, this is achieved through the use of wavefront sensors to detect laser and natural guide stars and deformable mirrors to direct the corrected wavefront to science instruments. These opto-mechanical designs and complex controls are constrained by several requirements that must be satisfied.



Figure 1. Thirty Meter Telescope. (Used with permission. Permission granted by Jamie Nakawatase.)

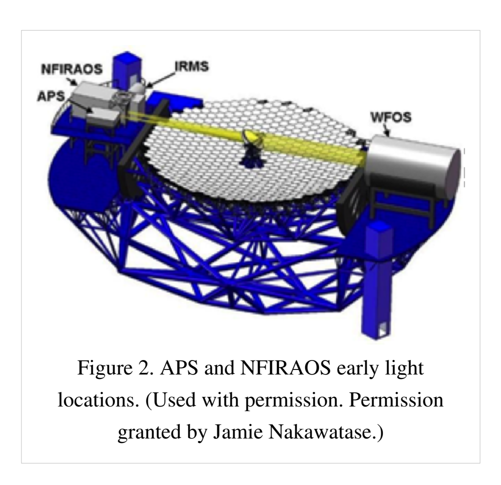


Figure 2. APS and NFIRAOS early light locations. (Used with permission. Permission granted by Jamie Nakawatase.)

TMT International Observatory (TIO), LLC is a non-profit organization of international members, responsible for managing the design, development, and operations of the TMT. The Jet Propulsion Laboratory (JPL) [9] participates in the design and development of several TMT subsystems and delivers an operational APS, where TIO is responsible for providing requirements to JPL. The APS team applies an MBSE approach to analyze requirements, derive an architecture design, and implement a system. TIO also works with JPL to analyze the operational behavior of NFIRAOS through modeling system-level operational scenarios (such as slew, acquisition, and dithering) with Monte-Carlo simulations. Modeling patterns are used to capture functional and physical system characteristics, behavior, requirements, parametric relationships, and use case scenarios. MBSE applications are motivated by optimization to better understand TMT’s complex system behaviors.

Purpose

This article describes how MBSE is applied to the development of critical subsystems of a complex interdisciplinary system and the benefits of this approach. While document-based artifacts are necessary throughout the development lifecycle, complex systems engineering relies significantly on the use of models to address concerns from various domains (e.g. mechanics, optics, controls) (Figure 3). MBSE helps to manage implicit dependencies on information contained in these cross-domain documents, understand change impacts, analyze designs, and communicate evolving technical baselines. Models act as the single source of authority for systems engineering information that enables optimization through consistent and automated data exchange, enhanced analyses, and consolidating subsystem design information into separate artifacts needed by various stakeholders.

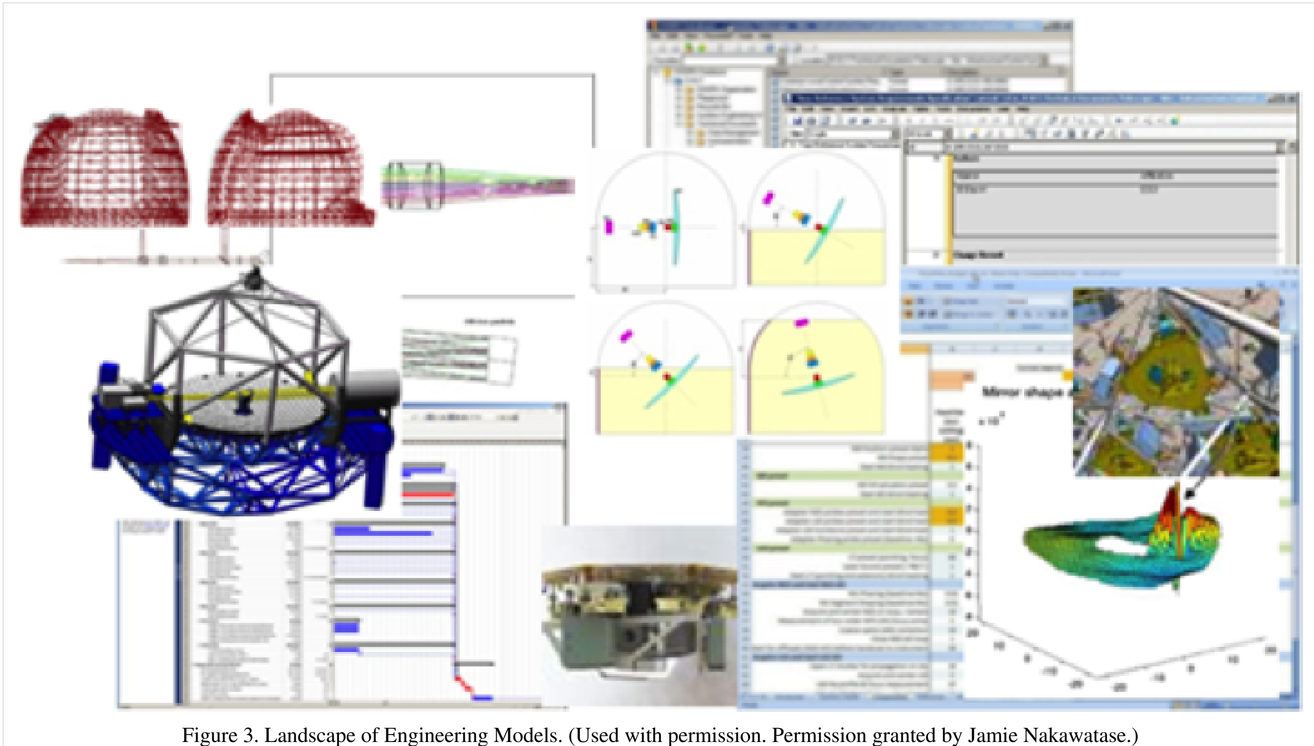


Figure 3. Landscape of Engineering Models. (Used with permission. Permission granted by Jamie Nakawatase.)

MBSE Challenges

Systems engineering (SE) is inherently challenging with concepts spanning several engineering disciplines. Expressing these concepts in a model demands the use of flexible methodology, language, and tools. The modeler must understand how to leverage this flexibility to rigorously specify systems with a broad range of complex design considerations. This poses the initial MBSE challenge—**the learning curve**. However, this challenge is native to any new undertaking and is overcome by hands-on experience, training, and ever-growing resources.

Another challenge is determining how to **apply** MBSE to meet the needs of a project. MBSE is not separate from SE—it is an approach to achieve the fundamental goals of SE more efficiently. MBSE should not be used to duplicate work, but instead replace efforts that can be better accomplished formally through modeling.

Engineers live in a landscape of variability among tools and information models in different domains (e.g. ALM, PLM, CAD), which are often implicitly connected [4]. To benefit from a model-based paradigm, the models require explicit connections. A key challenge is how to leverage data and associated models to enable **cross-domain integration**.

Finally, **standardization** is a challenge for MBSE. A model is only effective if stakeholders can understand it. SysML is an evolving modeling language that is becoming the dominant standard to communicate system architectures, independent of the modeling tools that use it. However, SysML is only one of many standards that must be applied to ensure the models can be consistently interpreted by tools and users.

MBSE Approach

The MBSE approach follows the conventional SE V-process enriched by a model-based paradigm (Figure 4). The scope of this MBSE application addresses models at L2 and L3, and requirements flow-down to L4, although the approach can be applied to support specification and architecture design at other levels as well. Associated modeling artifacts are created early and maintained throughout the development lifecycle.

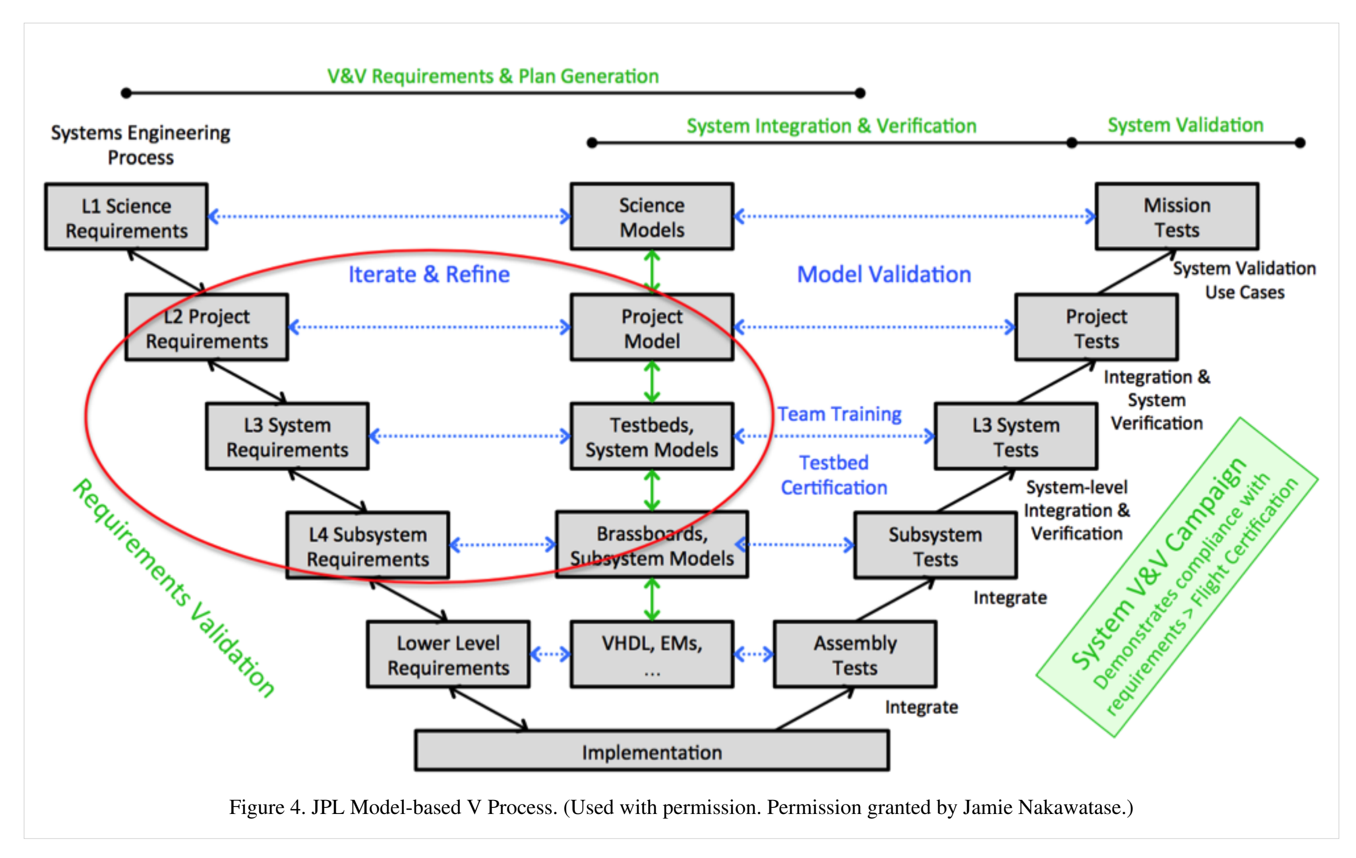


Figure 4. JPL Model-based V Process. (Used with permission. Permission granted by Jamie Nakawatase.)

MBSE articulates the system architecture in a formal, executable model that captures structure, behavior, requirements, and parametric relationships of system elements. Operational scenarios are defined in the model and analyzed through system-level simulation to verify requirements and validate overall system design over an expected range of conditions and parameters. The system model is also the authoritative source for several engineering documents. The MBSE approach is subsequently described by its methodology, tooling infrastructure, and analysis processes.

Methodology

The Executable Systems Engineering Method (ESEM) [1] is used to formalize requirements, specify system designs, characterize components, and specify/run analyses. ESEM augments the Object Oriented Systems Engineering Method (OOSEM) [2] by enabling executable models that enhance understanding, precision, and verification of requirements through applying analysis patterns specified with various SysML diagrams. ESEM also enables integration of supplier/customer models.

Figure 5 shows the major activities common to SE processes using OOSEM. The red circles indicate where ESEM injects formal modeling methods.

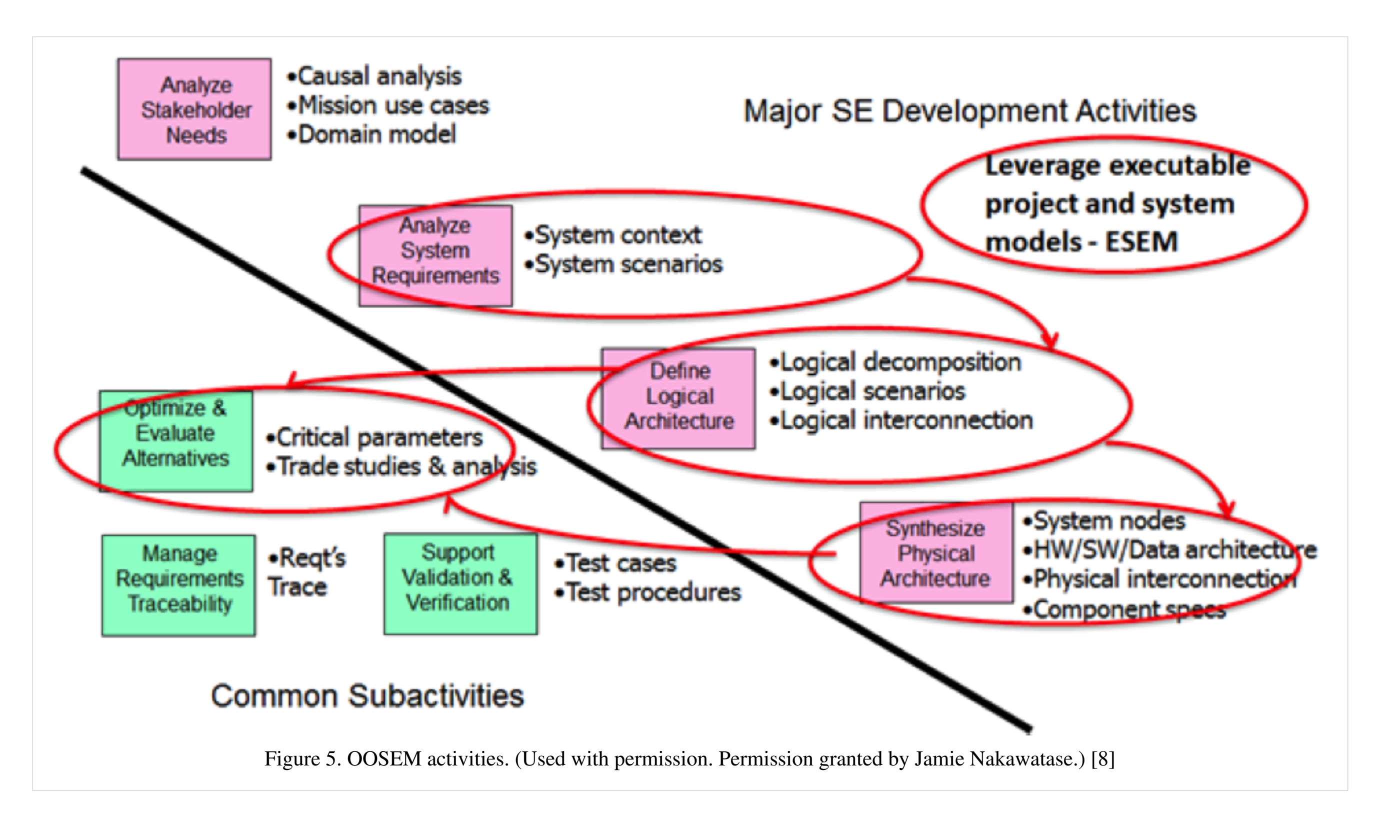


Figure 5. OOSEM activities. (Used with permission. Permission granted by Jamie Nakawatase.) [8]

ESEM is utilized to model different levels of abstraction that are analyzed using several modeling patterns as detailed in [4]. The system-of-interest is modeled as a black box that interacts with external subsystems, such as controls. Interactions are modeled using ports to identify operations and flows at the system-of-interest interface.

The conceptual model specifies technology-independent system components and captures their behavior. This part of the model is used to analyze characteristics such as duration of operational scenarios. Component behavior is captured using state machines and activity diagrams, and constraint parameters are captured in a table. Communication across internal and external system components is accomplished through the sending and receiving of signals through ports. This model supports production of interface control documents by querying information sent from one component to another over ports.

The conceptual model serves as a basis to specify the realization model of the physical components. The realization model imposes technology-dependent constraints on the design solutions. Both the conceptual (i.e. logical) and realization (i.e. physical) models represent the “as-specified” system.

Tooling

The MBSE approach uses standard SysML language and modeling tools to minimize custom software. The OpenMBEE community promotes an open tooling environment that provides a platform for modeling. It utilizes the Model Management System (MMS) that can be accessed from rich SysML desktop clients like MagicDraw, lightweight web-based clients like View Editor, computational programs like Mathematica, and other tools that utilize RESTful web services. The MMS also provides the basic infrastructure for search, relation management, versioning, workflow, access control, content flexibility, web applications support, web-based API access, and multi-tool/repository integration across engineering and management disciplines.

Figure 6 shows the integration of model artifacts produced by MMS, View Editor, and MagicDraw. System models are constructed, queried, and rendered following the view and viewpoint paradigm [3] from MMS.

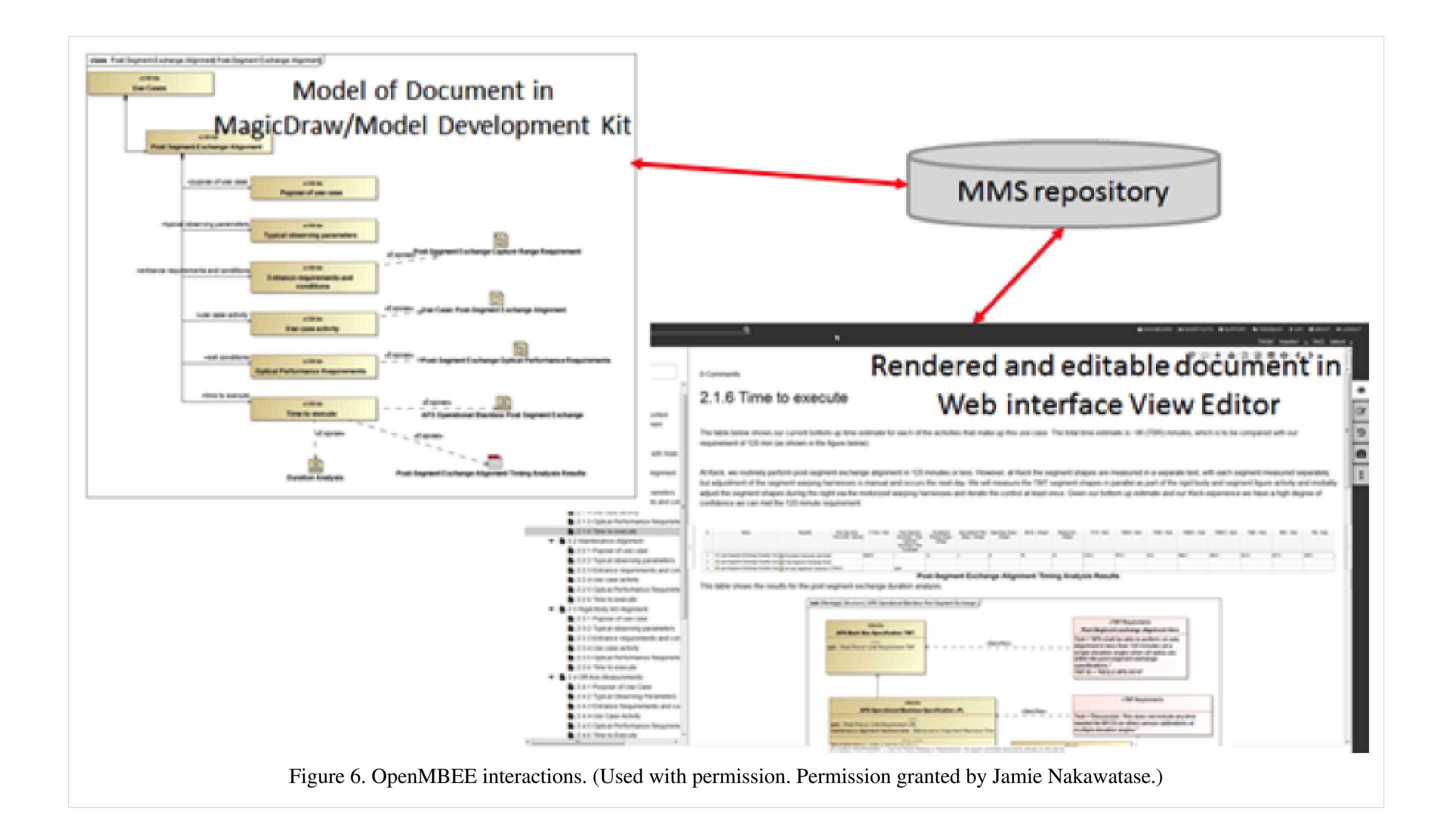


Figure 6. OpenMBEE interactions. (Used with permission. Permission granted by Jamie Nakawatase.)

Analysis

One key SE process is to analyze the impact of changing requirements on the system design. Figure 7 illustrates how the MBSE approach is used to support requirements impact analysis through the following steps:

- Step 1: A changed requirement triggers impact analysis.
- Step 2: MMS integrates DOORS (which manages text requirements) and the SysML model, enabling a DOORS requirement change to propagate to the SysML model.
- Step 3: A property-based requirement is formalized in SysML, enabling requirements specification that can be evaluated by engineering analysis.
- Steps 4-6: The conceptual and/or realization design is automatically verified against the changed requirement, resulting in pass or fail.

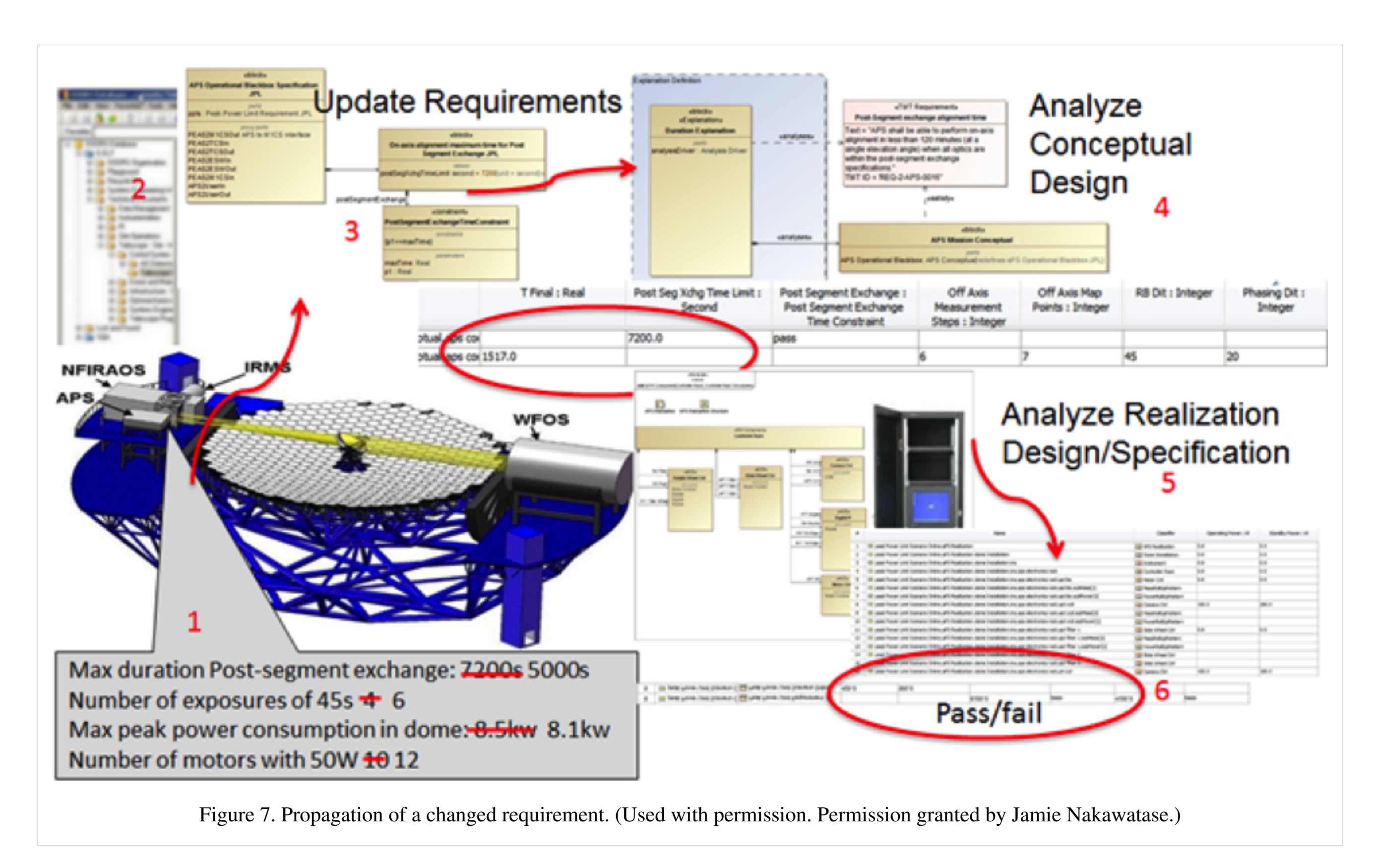


Figure 7. Propagation of a changed requirement. (Used with permission. Permission granted by Jamie Nakawatase.)

In this article, a simplified APS model is shown to illustrate the model-based approach for analyzing changed requirements. The full analysis is available in [4] and [5].

A model-based approach was also applied to APS for error analysis, which was performed to describe the accuracy of expected system performance against requirements. It involved multiple artifacts to analyze a requirement such as "APS shall measure the position of the telescope pupil to an accuracy of 0.03% of the diameter of the pupil." Defining requirements and parameters in the model indicated the required accuracy and current best estimates of the system design. Defining various roll-up patterns allowed for error decomposition calculations. The benefit is realized in automated requirements verification when applying a parametric solver to formulate results for specified equations in the model. This model-based approach formally integrates accuracy requirements with the system design.

The system model for NFIRAOS LGS MCAO and NGSAO modes was developed to capture sequence behaviors and operational scenarios to run Monte-Carlo simulations for verifying acquisition time, observing efficiency, and operational behavior requirements. The model is particularly useful for investigating the effect of parallelization, identifying interface issues, and re-ordering sequence acquisition tasks.

ESEM enables system analysis by conducting quantitative assessments to select and/or update the most efficient system architecture and generate derived engineering data. System analysis provides rigor for technical decision-making. It includes modeling and simulation, cost, technical risks, and effectiveness analyses, and is used to perform trade studies. In particular, it supports requirements verification, which assesses whether a system design meets its objectives and satisfies the constraints levied by system requirements.

Observed Benefits

The MBSE approach applied to APS and NFIRAOS was motivated by optimization to coordinate the efforts of complex system development. In these applications, implicit dependencies are made explicit in a formal model through the use of ESEM, OpenMBEE, and SysML modeling constructs. Requirements are formalized and tracked directly to the evolving system design. This tight association of requirements within a common environment promotes cross-domain integration and efficient communication among stakeholders. The model is used to automate requirements verification and to generate systems engineering products. The benefit over a traditional document-based approach is that currently disconnected artifacts become related in the model, enabling the production of consistent model-based documentation. Requirements verification is an important analysis conducted in the context of MBSE. To perform this analysis, the requirements, executable behavior, and models predicting the system's performance must be integrated. The ability to integrate these elements using ESEM and the OpenMBEE tooling infrastructure is a significant value proposition for the MBSE approach described in this article. In the formally integrated and executable SysML model, simulations are performed to analyze the impact of changed requirements and verify that requirements are met within specified constraints for various operational scenarios. MBSE enhances information exchange through created visualizations that communicate system behavior. For example, duration analyses were performed to study acquisition time for observing. The use of Monte-Carlo simulations proved how the model-based approach optimized the analysis process. Higher quality analysis results were obtained through the execution of operational scenario runs in an articulated system model, and the model continues to serve as a communication tool across various domains.

Acknowledgements

This research was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA), and at NoMagic. The TMT Project gratefully acknowledges the support of the TMT collaborating institutions. They are the California Institute of Technology, the University of California, the National Astronomical Observatory of Japan, the National Astronomical Observatories of China and their consortium partners, the Department of Science and Technology of India and their supported institutes, and the National Research Council of Canada. This work was supported as well by the Gordon and Betty Moore Foundation, the Canada Foundation for Innovation, the Ontario Ministry of Research and Innovation, the Natural Sciences and Engineering Research Council of Canada, the British Columbia Knowledge Development Fund, the Association of Canadian Universities for Research in Astronomy (ACURA), the Association of Universities for Research in Astronomy (AURA), the U.S. National Science Foundation, the National Institutes of Natural Sciences of Japan, and the Department of Atomic Energy of India.

References

Works Cited

[1] Zwemer, D., "Connecting SysML with PLM/ALM, CAD, Simulation, Requirements, and Project Management Tools", Intercax LLC, May 2011. [2] Friedenthal S, Moore A., and Steiner R., "A Practical Guide to SysML 3rd Ed.", Morgan Kaufmann OMG Press, 2014 [3] ISO/IEC, ISO/IEC 42010:2011, Systems and software engineering - Architecture description

Primary References

[4] Karban, R., Jankevičius, N., Elaasar, M. "ESEM: Automated Systems Analysis using Executable SysML Modeling Patterns", INCOSE International Symposium (IS), Edinburgh, Scotland, 2016 [5] Karban, R. "Using Executable SysML Models to Generate System Engineering Products", NoMagic World Symposium, 2016 [6] OMG SysML, "Systems Modeling Language (SysML) Version 1.4", OMG, 2014 [7] Open Source Engineering

Environment: https://open-mbee.github.io/[8] TMT, “Thirty Meter Telescope.” http://www.tmt.org [9] JPL, “Jet Propulsion Laboratory.” https://www.jpl.nasa.gov/[10] Karban R., Dekens F., Herzig S., Elaasar M, Jankevičius N., “Creating systems engineering products with executable models in a model-based engineering environment”, SPIE, Edinburgh, Scotland, 2016

Additional References

https://github.com/Open-MBEE/TMT-SysML-Model https://github.com/Open-MBEE/mdk/tree/support/2.5/manual http://www.omgwiki.org/MBSE/doku.php?id=mbse:telescope https://groups.google.com/forum/#!forum/openmbee

---

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.9.1, released 16 October 2018

Miniature Seeker Technology Integration Spacecraft

The Miniature Seeker Technology Integration (MSTI) spacecraft was the first of its kind: a rapid development spacecraft, designed and launched in one year. As an aerospace example for a satellite application, the case study, "M.S.T.I.: Optimizing the Whole System" (Grenville, Kleiner, and Newcomb 2004), describes the project's systems engineering approach. Driven by an aggressive schedule, MSTI optimized over the whole project, rather than allowing sub-optimizations at the component level. As a partnership with Phillips Laboratories, the Jet Propulsion Laboratory (JPL), and Spectrum Astro, MSTI went into orbit on November 21, 1992. The MSTI-1 succeeded in meeting all primary mission objectives, surpassing the 6-day data collection mission requirement.

Domain Background

There are many case study examples for aerospace systems. This case is of particular interest because it highlights mechanisms which enabled successful performance following an aggressive schedule. Since this rapid development spacecraft was designed and launched in one year, new ways of structuring the project were necessary. Within this domain, the MSTI project used an innovative approach. Practices from this project led to the Mission Design Center and the System Test Bed at JPL.

Case Study Background

This case study was developed in support of the National Aeronautics and Space Administration (NASA) Program and Project Management Initiative by authors at the Virginia Polytechnic Institute and State University and Scientific Management, Inc. The case study was developed in the interest of continuously improving program and project management at NASA (NASA 2010). Research for this case included comprehensive literature review and detailed interviews. The project was selected based on the potential for providing lessons learned.

Case Study Description

The MSTI case study illustrates many principles described in the Systems Engineering Body of Knowledge (SEBoK). The MSTI team had to make adjustments to the traditional approach to spacecraft development in order to stay within budget and to meet the aggressive timeline of bringing a spacecraft from conception to launch within one year. The team realized that they were "building Porsches not Formula 1s"(Grenville, Kleiner, Newcomb 2004). Meeting the schedule was a crucial factor that affected all decisions. The SEBoK knowledge area on life cycle

models describes life cycle design in more detail.

The team took advantage of existing hardware architectures for their architectural design to expedite the project. In addition, at each design phase, the whole system was optimized instead of optimizing subsystems, and the level of optimization at the subsystem level was reduced. A hardware-in-the-loop test bed was used throughout the project, which expedited system integration.

The schedule was maintained only at a high level in the project management office, and the costs were managed using a cost reporting technique for "cost to completion." Rather than report on past spending, the Responsible Engineering Authorities (REAs) were expected to continually evaluate their ability to complete their tasks within projected costs. Faster procurement was achieved using the Hardware Acquisition Team, where a technical team member was matched with a procurement representative for each design function. This pair wrote the specifications together and initiated the purchase requisitions.

From the organizational perspective, increased responsibility and accountability were given to each team member. Individuals took ownership of their work and the decision process was streamlined. The team made more "good decisions," rather than optimal decisions. The team was collocated, and daily meetings were used to assign daily tasks and keep the team focused on the launch. The standard Problem Failure Report (PFR) was streamlined and electronic reports provided snapshots of the resolved and outstanding PFRs. The report helped REAs stay on top of potential problem areas. REAs were responsible for looking forward on the project horizon and notifying the team of any potential problem areas.

The first satellite in the MSTI series, MSTI-1, was launched on November 21, 1992. The spacecraft weighed 150 kg and was built for $19M in less than 12 months. Over 200,000 photographs were returned from the spacecraft. From a project management standpoint, all mission objectives were completed.

In addition, MSTI had a lasting legacy. Faster procurement developed into an approach JPL now calls "Fast Track Procurement." Hardware acquisition teams are used often in JPL projects. The hardware-in-the-loop test bed was the precursor to the Flight System Test Bed at JPL. Team members moved up quickly in JPL due to the increased responsibility and authority they were given on the MSTI project.

Summary

MSTI demonstrated that an aggressive schedule can be used to design low earth-orbiting spacecraft to optimize the full system. The MSTI experience changed JPL's culture and their approach to spacecraft development and mission management. The insights from this case study example can help both students and practitioners better understand principles described in the SEBoK.

References

Works Cited

Grenville, D., B.M. Kleiner, and J.F. Newcomb. 2004. *M.S.T.I., Optimizing the Whole System.* Blacksburg, VA: Virginia Polytechnic Institute, case study developed in support of the NASA Program and Project Management Initiative. 1-27. Accessed June 3, 2011. Available at http:/ / www. nasa. gov/ pdf/ 293212main_58529main_msti_casestudy_042604.pdf.

NASA. 2010. *A Catalog of NASA-Related Case Studies,* version 1.6. Compiled by the Office of the Chief Knowledge Officer, Goddard Space Flight Center, MD, USA: National Aeronautics and Space Administration (NASA). Accessed June 3, 2011. Available at http:/ / www. nasa. gov/ centers/ goddard/ pdf/ 450420main_NASA_Case_Study_Catalog.pdf.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.9.1, released 16 October 2018

Standard Korean Light Transit System

This example was created as a SE example directly for the SEBoK. It deals with systems engineering (SE) concepts and guidelines applied to the development of the Standard Korean Light Transit System (SKLTS). In Korea, local authorities had historically been interested in light transit to help resolve their transportation problems. The SKLTS was a joint effort between local authorities and the central government. It was built to provide a standard platform on which any local authority could construct its own light transit system. The issues of stakeholder requirements, safety, and reliability, availability, and maintainability were critical to the success of this system.

Description

The elements of the SKLTS were classified into four groups (as shown in Figure 1): trains, signal systems, electric and machinery (E&M) systems, and structures. Trains and vehicles were to be automatically operated, without need for human operators. Operation systems and their interfaces were based on digital signals and communications. For SKLTS, SE-based design activities focused on reliability, availability, maintainability, and safety (RAMS), and were integrated into project management (PM) activities during all phases.

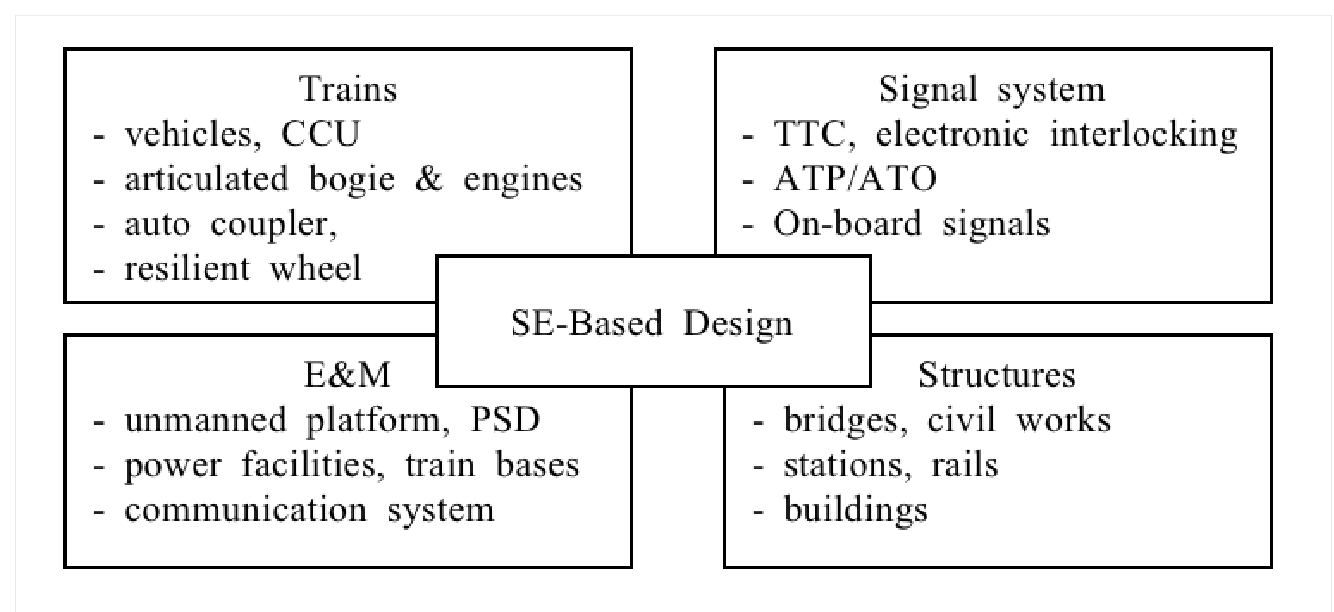


Figure 1. Subsystems of the SKLTS (Ahn, 2005). (Notes: CCU: Central Control Unit; TTC: Total Traffic Control; ATP: Automatic Train Protection; ATO: Automatic Train Operation; PSD: Platform Screen Door) Reprinted with permission of *Journal of the Korean Society for Railway*. All other rights are reserved by the copyright owner.

The project life cycle for the SKLTS is summarized in Figure 2. It consisted of 7 phases: concept studies, concept development, preliminary design, design, system production and testing, performance evaluation, and operation/maintenance/close-out (OMC) - please see (Choi 2007) and (Chung et al. 2010) for further details. These

phases, with the exception of the production and test phases, are completed through an evaluation and decision point (EDP) (milestone), depicted as a colored circle in Figure 2. These EDPs correspond to common life cycle artifacts such as requests for proposal (RFPs), proposals, preliminary design reviews (PDRs), and critical design reviews (CDRs).

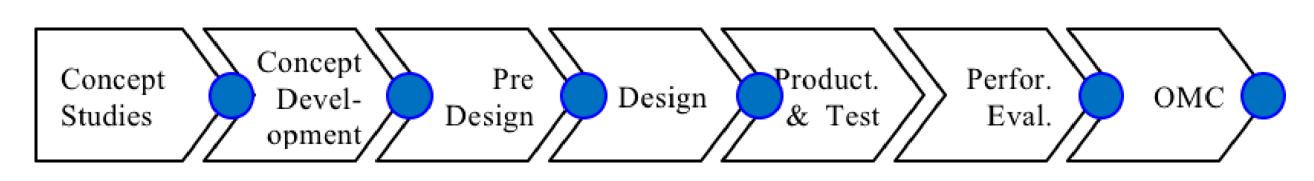


Figure 2. 7 phases of the SKLTS development (Ahn 2005). Reprinted with permission of the *Journal of the Korean Society for Railway*. All other rights are reserved by the copyright owner.

During the SKLTS development, SE activities were focused on RAMS as summarized in Table 1.

Table 1. The SE Framework of the SKLTS (Ahn 2005). Reprinted with permission of the *Journal of the Korean Society for Railway*. All other rights are reserved by the copyright owner.

Phases	Safety	Reliability	Function	Performance
Concept studies	• Requirements analysis	• Identifying RAM conditions • RAM allocation	• System configuration • Interface management	• Performance simulation
Concept development & pre-design	• Safety planning • Defining safety procedures & levels	• RAM planning • Initial availability analysis	• Defining scenarios and alarm procedure • Pre-designing command rooms	• Interface analysis
Design	• Hazard log • Safety case analysis • Risk analysis	• Reporting RAM analysis • RAM analysis of auxiliary systems	• Defining alarm systems • Train analysis • Functionality analysis of stations	• Interface analysis
Performance evaluation	• Safety test planning & testing	• Verification and Validation (V&V) RAM • Maintainability test	• System test planning and testing	• Performance test planning & testing
Initial Operation	• System acceptance • Driver certification	• RAM monitoring • FRACAS*	• Analyzing systems • Identifying improvement points	• Performance monitoring

*FRACAS: Failure Reporting & Corrective Action System

In the "concept studies" and "concept development" phases, requirements included the RAMS objectives. Planning activities in this phase included the scheduling of various tests and evaluations to be conducted after system design. The basic layout of rails and command rooms was also proposed. Finally, it was during this phase that interface management procedures and relationships between requirements and systems were defined. For RAMS engineering, it was also important to establish associated plans and criteria (e.g., RAM plans, safety plans, service availability, etc.).

During the pre-design phase, the basic architecture of the system was determined for safety planning, RAMS planning, and operational scenarios. Interfaces among subsystems were defined as well as management procedures for contractors and legal regulations. The functional analysis dealt with timeline, accuracy of stop points, and trip times. Pre-design activities also included the specifications of major system elements such as signal systems, trains, and interfaces. For RAMS engineering, safety scenarios were defined, and the hazard and risk analyses were

performed.

During the design and performance evaluation phases, hazard log and RAMS analyses were performed to ensure that each subsystem met safety requirements. The specifications of alarm systems and stations were also defined. In addition, V&V and test procedures were determined for performance evaluation. During the design phase, a design/construction interface manual (D/CIM) was developed and applied to ensure integrated and consistent design. (Bombardier, 2005)

Because SKLTS was designed as an automatically-driven system, RAMS issues were critical to its success. The safety and reliability of the SKLTS were evaluated on a test railway that was constructed to standard specifications. Data was gathered from existing Korean light rail systems, as well as the light rail systems from other countries, to support V&V activities.

Various methods were applied for achieving the RAMS objectives, including RAMS requirements analysis, safety and RAMS planning, utilization of systems scenarios, and construction risk analysis.

Initial operation of SKLTS was allowed only after the system was formally accepted and operators were properly certified. During test operation, RAMS performance was continuously monitored and system scenarios were used successfully to evaluate the dynamic behavior of the system. A failure reporting and corrective action system (FRACAS) was used to gather accident and failure data. Continuous improvement when the system is in normal operation was identified as a requirement; the results from the FRACAS will be used to support improvement of the system, maintenance, and improvement of procedures.

Summary

Korean local authorities have successfully introduced the SKLTS to their precincts with some modifications. Successful examples include the Inchun Airport Line and the Seoul 9th Subway Line. One lesson learned identified was that requirement analysis, especially in the first few phases, should have been more complete.

References

Works Cited

Ahn, S.H. 2005. “Systems Engineering Applied to the Construction of Unmanned Light Rail Transit Systems.” *Journal of the Korean Society for Railway.* 8(2): 41-49.

Bombardier. 2005. *Design/Construction Interface Manual*. Montréal, Québec, Canada: Bombardier.

Choi, Y.C. 2007. "Systems Engineering Application Model for the National R&D Project: Focusing on the Railway Systems." Ph.D. dissertation, Ajou University, 2007.

Chung, S. Y., S.G. Lee, D.W. Lee, and S.T. Jeon. 2010. “A Study on The Application Model of Systems Engineering to Advance the Business of the Light Rail Transit (LRT).” Proceedings on the Autumn Conference of the Korean Society for Railway, p. 24-30.

Primary References

None.

Additional References

Han, S.Y. and A.H. Lee. 2005. *System Engineering for The Technology Development Program of Light Transit Systems: Internal Research Report.* Gyeongi-do, Korea: Korea Railroad Research Institute.

Korean Agency for Technology and Standards. 2009. *KSX ISO/IEC 15288: Life Cycle Processes of IT Systems Based on Systems and Software Engineering, Research Report.* Gyeonggi-do, Korea: Korean Agency for Technology and Standards.

Lee, W.D. 2002. “A Study on the Top-Level Functional Analysis for the Automated Guideway Transit by System Engineering Tools.” Proceedings of the Autumn Conference of the Korean Society for Railway, p. 202-207.

< Previous Article | Parent Article | Last Article (Return to TOC)

SEBoK v. 1.9.1, released 16 October 2018

Article Sources and Contributors

Letter from the Editor *Source*: https://www.sebokwiki.org/d/index.php?oldid=53272 *Contributors*: Apyster, Bkcase, Cnielsen, Dholwell, Eleach, Kguillemette, Radcock, Rcloutier, Smenck2, Wikiexpert

BKCASE Governance and Editorial Board *Source*: https://www.sebokwiki.org/d/index.php?oldid=54206 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Kguillemette, Radcock, Rcloutier, Smenck2

Acknowledgements and Release History *Source*: https://www.sebokwiki.org/d/index.php?oldid=54531 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Ddori, Dhenry, Dholwell, Dnewbern, Eleach, Janthony, Jgercken, Kguillemette, Mhenshaw, Nicole.hutchison, Radcock, Smenck2, Wikiexpert

SEBoK Introduction *Source*: https://www.sebokwiki.org/d/index.php?oldid=54248 *Contributors*: Afaisandier, Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, HP.deKoning, Janthony, Jgercken, Kguillemette, Mhenshaw, Nicole.hutchison, Radcock, Smenck2, Wikiexpert, Zamoses

Introduction to the SEBoK *Source*: https://www.sebokwiki.org/d/index.php?oldid=54265 *Contributors*: Bkcase, Radcock

Scope of the SEBoK *Source*: https://www.sebokwiki.org/d/index.php?oldid=54195 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Groedler, Janthony, Jgercken, Mhenshaw, Nicole.hutchison, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Structure of the SEBoK *Source*: https://www.sebokwiki.org/d/index.php?oldid=54268 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Gparnell, Janthony, Jgercken, Nicole.hutchison, Radcock, Sfriedenthal, Skmackin, Smenck2, Wikiexpert, Zamoses

Introduction to Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54201 *Contributors*: Bkcase, Radcock

Systems Engineering Overview *Source*: https://www.sebokwiki.org/d/index.php?oldid=54203 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Radcock, Smenck2

Economic Value of Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54243 *Contributors*: Apyster, Asofer, Bkcase, Cnielsen, Dhenry, Dholwell, Eleach, Gparnell, Janthony, Mhenshaw, Radcock, Rvalerdi, Smenck2, Wikiexpert

Systems Engineering: Historic and Future Challenges *Source*: https://www.sebokwiki.org/d/index.php?oldid=54273 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Janthony, Jgercken, Radcock, Smenck2, Wikiexpert

Systems Engineering and Other Disciplines *Source*: https://www.sebokwiki.org/d/index.php?oldid=54275 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Janthony, Jgercken, Kguillemette, Nicole.hutchison, Skmackin, Smenck2, Wikiexpert, Zamoses

Introduction to SE Transformation *Source*: https://www.sebokwiki.org/d/index.php?oldid=54276 *Contributors*: Bkcase, Radcock

Transitioning Systems Engineering to a Model-based Discipline *Source*: https://www.sebokwiki.org/d/index.php?oldid=54279 *Contributors*: Bkcase, Radcock

Systems Engineering Core Concepts *Source*: https://www.sebokwiki.org/d/index.php?oldid=54282 *Contributors*: Bkcase

SEBoK Users and Uses *Source*: https://www.sebokwiki.org/d/index.php?oldid=54293 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Jgercken, Kguillemette, Nicole.hutchison, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Use Case 0: Systems Engineering Novices *Source*: https://www.sebokwiki.org/d/index.php?oldid=54295 *Contributors*: Bkcase, Cnielsen, Dholwell, Radcock

Use Case 1: Practicing Systems Engineers *Source*: https://www.sebokwiki.org/d/index.php?oldid=54297 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Gparnell, Janthony, Jgercken, Radcock, Rmadachy, Smenck2, Wikiexpert

Use Case 2: Other Engineers *Source*: https://www.sebokwiki.org/d/index.php?oldid=54299 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Jgercken, Rmadachy, Smenck2, Wikiexpert

Use Case 3: Customers of Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54301 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Gparnell, Jgercken, Radcock, Rmadachy, Smenck2, Wikiexpert

Use Case 4: Educators and Researchers *Source*: https://www.sebokwiki.org/d/index.php?oldid=54303 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Jgercken, Mhenshaw, Radcock, Rmadachy, Smenck2, Wikiexpert

Use Case 5: General Managers *Source*: https://www.sebokwiki.org/d/index.php?oldid=54304 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Gparnell, Jgercken, Mhenshaw, Rmadachy, Smenck2, Wikiexpert

Foundations of Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54310 *Contributors*: Alee, Apyster, Asquires, Bkcase, Cdagli, Cnielsen, Dcarey, Ddori, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhenshaw, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Systems Fundamentals *Source*: https://www.sebokwiki.org/d/index.php?oldid=54312 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Janthony, Mhenshaw, Radcock, Smenck2, Wikiexpert, Ymordecai

What is a System? *Source*: https://www.sebokwiki.org/d/index.php?oldid=54315 *Contributors*: Afaisandier, Alee, Apyster, Asquires, Bkcase, Bwells, Ccalvano, Cnielsen, Ddori, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Types of Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54316 *Contributors*: Apyster, Asquires, Bkcase, Bwells, Ccalvano, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Jsnoderly, Radcock, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Groupings of Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54317 *Contributors*: Apyster, Asquires, Bkcase, Bwells, Ccalvano, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Complexity *Source*: https://www.sebokwiki.org/d/index.php?oldid=54319 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Jsnoderly, Radcock, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Emergence *Source*: https://www.sebokwiki.org/d/index.php?oldid=54322 *Contributors*: Apyster, Bkcase, Cnielsen, Dfairley, Dhenry, Janthony, Jgercken, Mhenshaw, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Systems Science *Source*: https://www.sebokwiki.org/d/index.php?oldid=54327 *Contributors*: Apyster, Asquires, Bkcase, Blawson, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

History of Systems Science *Source*: https://www.sebokwiki.org/d/index.php?oldid=54328 *Contributors*: Apyster, Asquires, Bkcase, Ccalvano, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Qwang, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Systems Approaches *Source*: https://www.sebokwiki.org/d/index.php?oldid=54329 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Mhenshaw, Radcock, Smenck2, Wikiexpert

Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=54331 *Contributors*: Apyster, Asquires, Bkcase, Blawson, Ccalvano, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

What is Systems Thinking? *Source*: https://www.sebokwiki.org/d/index.php?oldid=54332 *Contributors*: Apyster, Asquires, Bkcase, Blawson, Bwells, Ccalvano, Cnielsen, Dhenry, Dnewbern, Janthony, Jgercken, Radcock, Rturner, Sjackson, Smenck2, Wikiexpert

Concepts of Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=54333 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Mhenshaw, Radcock, Sfriedenthal, Sjackson, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Principles of Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=54334 *Contributors*: Bkcase, Bsauser, Cnielsen, Dhenry, Dholwell, Janthony, Mhenshaw, Radcock, Smenck2, Wikiexpert, Ymordecai

Patterns of Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=54336 *Contributors*: Bkcase, Cnielsen, Dhenry, Dnewbern, Janthony, Mhenshaw, Radcock, Smenck2

Representing Systems with Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=54337 *Contributors*: Apyster, Bkcase, Cnielsen, Ddori, Dhenry, Dholwell, Dnewbern, Eleach, Gparnell, Janthony, Jgercken, Radcock, Sfriedenthal, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

What is a Model? *Source*: https://www.sebokwiki.org/d/index.php?oldid=54339 *Contributors*: Alee, Apyster, Bkcase, Cnielsen, Ddori, Dhenry, Gparnell, Janthony, Jgercken, Mhenshaw, Radcock, Sfriedenthal, Skmackin, Smenck2, Wikiexpert, Zamoses

Why Model? *Source*: https://www.sebokwiki.org/d/index.php?oldid=54340 *Contributors*: Alee, Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Janthony, Jgercken, Radcock, Sfriedenthal, Skmackin, Smenck2, Wikiexpert, Zamoses

Types of Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=54341 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Gparnell, Janthony, Jgercken, Mhenshaw, Radcock, Sfriedenthal, Skmackin, Smenck2, Wikiexpert, Zamoses

System Modeling Concepts *Source*: https://www.sebokwiki.org/d/index.php?oldid=54342 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Ddori, Dhenry, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Sfriedenthal, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Integrating Supporting Aspects into System Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=54344 *Contributors*: Bkcase, Cnielsen, Eleach, Smenck2, Ymordecai

Modeling Standards *Source*: https://www.sebokwiki.org/d/index.php?oldid=54345 *Contributors*: Apyster, Bkcase, Cnielsen, Dcarey, Ddori, Dhenry, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Sfriedenthal, Skmackin, Smenck2, Wikiexpert, Zamoses

Systems Approach Applied to Engineered Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54347 *Contributors*: Apyster, Bkcase, Blawson, Cnielsen, Dcarey, Dhenry, Dholwell, Dnewbern, Eleach, Gparnell, Janthony, Jgercken, Mhenshaw, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Overview of the Systems Approach *Source*: https://www.sebokwiki.org/d/index.php?oldid=54350 *Contributors*: Apyster, Bkcase, Blawson, Bwells, Cnielsen, Dcarey, Dhenry, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Sfriedenthal, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Engineered System Context *Source*: https://www.sebokwiki.org/d/index.php?oldid=54353 *Contributors*: Apyster, Asquires, Bkcase, Bwells, Ccalvano, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Identifying and Understanding Problems and Opportunities *Source*: https://www.sebokwiki.org/d/index.php?oldid=54356 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Rturner, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Synthesizing Possible Solutions *Source*: https://www.sebokwiki.org/d/index.php?oldid=54357 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Rturner, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Analysis and Selection between Alternative Solutions *Source*: https://www.sebokwiki.org/d/index.php?oldid=54254 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Rturner, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Implementing and Proving a Solution *Source*: https://www.sebokwiki.org/d/index.php?oldid=54359 *Contributors*: Apyster, Bkcase, Bwells, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Deploying, Using, and Sustaining Systems to Solve Problems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54362 *Contributors*: Apyster, Bkcase, Bwells, Cnielsen, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Mhenshaw, Radcock, Rturner, Sfriedenthal, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Stakeholder Responsibility *Source*: https://www.sebokwiki.org/d/index.php?oldid=54364 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Eleach, Gparnell, Janthony, Mhenshaw, Radcock, Smenck2, Wikiexpert, Ymordecai

Applying the Systems Approach *Source*: https://www.sebokwiki.org/d/index.php?oldid=54300 *Contributors*: Alee, Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Hdavidz, Janthony, Jgercken, Mhenshaw, Radcock, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Systems Engineering and Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54365 *Contributors*: Afaisandier, Apyster, Bkcase, Blawson, Cnielsen, Dcarey, Dfairley, Dhenry, Dholwell, Dnewbern, Groedler, Janthony, Jgercken, Mhenshaw, Radcock, Rmadachy, Sfriedenthal, Skmackin, Smenck2, Wikiexpert, Zamoses

Introduction to Life Cycle Processes *Source*: https://www.sebokwiki.org/d/index.php?oldid=54367 *Contributors*: Bkcase, Radcock, Sfriedenthal

Generic Life Cycle Model *Source*: https://www.sebokwiki.org/d/index.php?oldid=54369 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Kforsberg, Mhenshaw, Radcock, Rturner, Skmackin, Wikiexpert, Zamoses

Applying Life Cycle Processes *Source*: https://www.sebokwiki.org/d/index.php?oldid=54370 *Contributors*: Bkcase, Radcock

Life Cycle Processes and Enterprise Need *Source*: https://www.sebokwiki.org/d/index.php?oldid=54373 *Contributors*: Bkcase, Radcock

Life Cycle Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=54171 *Contributors*: Afaisandier, Apyster, Asquires, Bkcase, Dcarey, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Kforsberg, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

System Life Cycle Process Drivers and Choices *Source*: https://www.sebokwiki.org/d/index.php?oldid=54377 *Contributors*: Apyster, Bkcase, Cnielsen, Dcarey, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Kforsberg, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

System Life Cycle Process Models: Vee *Source*: https://www.sebokwiki.org/d/index.php?oldid=54378 *Contributors*: Apyster, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Janthony, Jgercken, Kforsberg, Mhenshaw, Rmadachy, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

System Life Cycle Process Models: Iterative *Source*: https://www.sebokwiki.org/d/index.php?oldid=54379 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Kforsberg, Mhenshaw, Smenck2, Wikiexpert, Ymordecai

Integration of Process and Product Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=54382 *Contributors*: Apyster, Bkcase, Blawson, Dhenry, Dholwell, Jgercken, Kforsberg, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Lean Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54384 *Contributors*: Bkcase, Dhenry, Dholwell, Dnewbern, Smenck2, Wikiexpert

Concept Definition *Source*: https://www.sebokwiki.org/d/index.php?oldid=54278 *Contributors*: Bkcase, Dhenry, Dholwell, Eleach, Groedler, Mhenshaw, Radcock, Smenck2, Wikiexpert, Ymordecai

Business or Mission Analysis *Source*: https://www.sebokwiki.org/d/index.php?oldid=54386 *Contributors*: Afaisandier, Asquires, Bkcase, Dhenry, Dholwell, Groedler, Janthony, Jgercken, Mhenshaw, Radcock, Rturner, Sjackson, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Stakeholder Needs and Requirements *Source*: https://www.sebokwiki.org/d/index.php?oldid=54388 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Janthony, Mhenshaw, Radcock, Smenck2, Wikiexpert, Ymordecai

System Definition *Source*: https://www.sebokwiki.org/d/index.php?oldid=54394 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

System Requirements *Source*: https://www.sebokwiki.org/d/index.php?oldid=54392 *Contributors*: Afaisandier, Apyster, Asofer, Asquires, Bkcase, Cjones, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Radcock, Rturner, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

System Architecture *Source*: https://www.sebokwiki.org/d/index.php?oldid=54393 *Contributors*: Afaisandier, Bkcase, Radcock

Logical Architecture Model Development *Source*: https://www.sebokwiki.org/d/index.php?oldid=54397 *Contributors*: Afaisandier, Apyster, Asquires, Bkcase, Dhenry, Dholwell, Dnewbern, Eleach, Groedler, Jgercken, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Physical Architecture Model Development *Source*: https://www.sebokwiki.org/d/index.php?oldid=54399 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dnewbern, Eleach, Mhenshaw, Radcock, Smenck2, Wikiexpert

System Design *Source*: https://www.sebokwiki.org/d/index.php?oldid=54401 *Contributors*: Afaisandier, Bkcase, Radcock

System Analysis *Source*: https://www.sebokwiki.org/d/index.php?oldid=54403 *Contributors*: Afaisandier, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Mhenshaw, Radcock, Rmadachy, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

System Realization *Source*: https://www.sebokwiki.org/d/index.php?oldid=54404 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Jgercken, Jsnoderly, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

System Implementation *Source*: https://www.sebokwiki.org/d/index.php?oldid=54406 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Jgercken, Jsnoderly, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

System Integration *Source*: https://www.sebokwiki.org/d/index.php?oldid=54408 *Contributors*: Afaisandier, Bkcase, Dhenry, Dholwell, Dnewbern, Janthony, Jgercken, Jsnoderly, Mhenshaw, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

System Verification *Source*: https://www.sebokwiki.org/d/index.php?oldid=54410 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Jgercken, Jsnoderly, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

System Validation *Source*: https://www.sebokwiki.org/d/index.php?oldid=54412 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Mhenshaw, Radcock, Smenck2, Wikiexpert

System Deployment and Use *Source*: https://www.sebokwiki.org/d/index.php?oldid=54413 *Contributors*: Bkcase, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Jsnoderly, Sjackson, Skmackin, Wikiexpert, Zamoses

System Deployment *Source*: https://www.sebokwiki.org/d/index.php?oldid=54414 *Contributors*: Apyster, Bgallagher, Bkcase, Dhenry, Dholwell, Dnewbern, Jgercken, Mhenshaw, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Operation of the System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54416 *Contributors*: Apyster, Bgallagher, Bkcase, Dhenry, Dholwell, Dnewbern, Jgercken, Ldecardenas, Mhenshaw, Sjackson, Skmackin, Wikiexpert, Zamoses

System Maintenance *Source*: https://www.sebokwiki.org/d/index.php?oldid=54417 *Contributors*: Bgallagher, Bkcase, Ddorgan, Dhenry, Dholwell, Dnewbern, Jgercken, Sjackson, Skmackin, Wikiexpert, Zamoses

Logistics *Source*: https://www.sebokwiki.org/d/index.php?oldid=54419 *Contributors*: Bkcase, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Jsnoderly, Mhenshaw, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Systems Engineering Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54421 *Contributors*: Bkcase, Dhenry, Dholwell, Gparnell, Groedler, Jgercken, Mhenshaw, Rmadachy, Skmackin, Smenck2, Wikiexpert, Zamoses

Planning *Source*: https://www.sebokwiki.org/d/index.php?oldid=54422 *Contributors*: Apyster, Bkcase, Bwells, Dhenry, Dholwell, Dnewbern, Groedler, Janthony, Jgercken, Mhenshaw, Rmadachy, Skmackin, Smenck2, Wikiexpert, Zamoses

Assessment and Control *Source*: https://www.sebokwiki.org/d/index.php?oldid=54415 *Contributors*: Apickard, Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Mhenshaw, Rmadachy, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

Risk Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54424 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Dnewbern, Econrow, Groedler, Jgercken, Kguillemette, Mhenshaw, Rmadachy, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

Measurement *Source*: https://www.sebokwiki.org/d/index.php?oldid=54428 *Contributors*: Apyster, Bkcase, Cjones, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Mhenshaw, Rcarson, Rmadachy, Skmackin, Smenck2, Wikiexpert, Zamoses

Decision Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54407 *Contributors*: Asquires, Bkcase, Dhenry, Dholwell, Dnewbern, Eleach, Gparnell, Groedler, Jgercken, Mhenshaw, Rmadachy, Skmackin, Smenck2, Wikiexpert, Zamoses

Configuration Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54207 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Jsnoderly, Mhenshaw, Radcock, Rmadachy, Skmackin, Smenck2, Wikiexpert, Zamoses

Information Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54429 *Contributors*: Apickard, Bkcase, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Mhenshaw, Rmadachy, Skmackin, Smenck2, Wikiexpert, Zamoses

Quality Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54430 *Contributors*: Bkcase, Dfairley, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Mhenshaw, Mtowhid, Qwang, Skmackin, Smenck2, Wikiexpert, Zamoses

Product and Service Life Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54431 *Contributors*: Apyster, Bkcase, Bstiffler, Dhenry, Dholwell, Dnewbern, Jgercken, Mhenshaw, Skmackin, Wikiexpert, Zamoses

Service Life Extension *Source*: https://www.sebokwiki.org/d/index.php?oldid=54432 *Contributors*: Asquires, Bkcase, Bstiffler, Bwells, Dhenry, Dholwell, Dnewbern, Jgercken, Mhenshaw, Skmackin, Wikiexpert, Zamoses

Capability Updates, Upgrades, and Modernization *Source*: https://www.sebokwiki.org/d/index.php?oldid=54280 *Contributors*: Asquires, Bkcase, Bstiffler, Bwells, Dhenry, Dholwell, Dnewbern, Jgercken, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Disposal and Retirement *Source*: https://www.sebokwiki.org/d/index.php?oldid=54212 *Contributors*: Asquires, Bkcase, Bstiffler, Bwells, Dhenry, Dholwell, Jgercken, Mhenshaw, Skmackin, Wikiexpert, Ymordecai, Zamoses

Systems Engineering Standards *Source*: https://www.sebokwiki.org/d/index.php?oldid=54434 *Contributors*: Bkcase, Ccalvano, Dhenry, Dholwell, Dnewbern, Groedler, Jgercken, Mhenshaw, Skmackin, Wikiexpert, Zamoses

Relevant Standards *Source*: https://www.sebokwiki.org/d/index.php?oldid=54436 *Contributors*: Bkcase, Ccalvano, Dhenry, Dholwell, Dnewbern, Groedler, Janthony, Jgercken, Kguillemette, Kzemrowski, Mhenshaw, Sfriedenthal, Skmackin, Smenck2, Wikiexpert, Zamoses

Alignment and Comparison of the Standards *Source*: https://www.sebokwiki.org/d/index.php?oldid=54210 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Groedler, Hdavidz, Janthony, Jgercken, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Application of Systems Engineering Standards *Source*: https://www.sebokwiki.org/d/index.php?oldid=54343 *Contributors*: Bkcase, Blawson, Dhenry, Dholwell, Dnewbern, Groedler, Hdavidz, Janthony, Jgercken, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Applications of Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54225 *Contributors*: Bkcase, Blawson, Dcarey, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Radcock, Smenck2, Wikiexpert, Zamoses

Product Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54439 *Contributors*: Bkcase, Blawson, Dcarey, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Skmackin, Smenck2, Wikiexpert, Zamoses

Product Systems Engineering Background *Source*: https://www.sebokwiki.org/d/index.php?oldid=54441 *Contributors*: Bkcase, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Smenck2, Wikiexpert

Product as a System Fundamentals *Source*: https://www.sebokwiki.org/d/index.php?oldid=54442 *Contributors*: Bkcase, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Smenck2, Wikiexpert

Business Activities Related to Product Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54260 *Contributors*: Bkcase, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Rlpineda, Wikiexpert

Product Systems Engineering Key Aspects *Source*: https://www.sebokwiki.org/d/index.php?oldid=54445 *Contributors*: Bkcase, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Smenck2, Wikiexpert

Product Systems Engineering Special Activities *Source*: https://www.sebokwiki.org/d/index.php?oldid=54447 *Contributors*: Bkcase, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Smenck2, Wikiexpert

Service Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54449 *Contributors*: Apyster, Bkcase, Blawson, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Jmartin, Kguillemette, Mhenshaw, Rlpineda, Rturner, Smenck2, Wikiexpert, Zamoses

Service Systems Background *Source*: https://www.sebokwiki.org/d/index.php?oldid=54450 *Contributors*: Bkcase, Blawson, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Smenck2, Wikiexpert, Zamoses

Fundamentals of Services *Source*: https://www.sebokwiki.org/d/index.php?oldid=54451 *Contributors*: Bkcase, Blawson, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Smenck2, Wikiexpert, Zamoses

Properties of Services *Source*: https://www.sebokwiki.org/d/index.php?oldid=54453 *Contributors*: Asquires, Bkcase, Blawson, Bwells, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Rlpineda, Smenck2, Wikiexpert, Zamoses

Scope of Service Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54456 *Contributors*: Bkcase, Blawson, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Smenck2, Wikiexpert, Zamoses

Value of Service Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54457 *Contributors*: Bkcase, Blawson, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Mhenshaw, Rlpineda, Rturner, Smenck2, Wikiexpert, Zamoses

Service Systems Engineering Stages *Source*: https://www.sebokwiki.org/d/index.php?oldid=54459 *Contributors*: Asquires, Bkcase, Blawson, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Kguillemette, Rlpineda, Rturner, Smenck2, Wikiexpert, Zamoses

Enterprise Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54178 *Contributors*: Apyster, Bkcase, Blawson, Dfairley, Dhenry, Dholwell, Janthony, Jgercken, Jmartin, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Enterprise Systems Engineering Background *Source*: https://www.sebokwiki.org/d/index.php?oldid=54461 *Contributors*: Afaisandier, Bkcase, Blawson, Dfairley, Dhenry, Dholwell, Janthony, Jgercken, Jmartin, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert

The Enterprise as a System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54462 *Contributors*: Bkcase, Blawson, Dhenry, Dholwell, Janthony, Jdahmann, Jgercken, Jmartin, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Related Business Activities *Source*: https://www.sebokwiki.org/d/index.php?oldid=54463 *Contributors*: Apyster, Bkcase, Blawson, Dfairley, Dhenry, Dholwell, Janthony, Jgercken, Jmartin, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Enterprise Systems Engineering Key Concepts *Source*: https://www.sebokwiki.org/d/index.php?oldid=54464 *Contributors*: Afaisandier, Bkcase, Blawson, Dhenry, Dholwell, Janthony, Jgercken, Jmartin, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Enterprise Systems Engineering Process Activities *Source*: https://www.sebokwiki.org/d/index.php?oldid=54465 *Contributors*: Afaisandier, Asquires, Bkcase, Blawson, Dhenry, Dholwell, Janthony, Jgercken, Jmartin, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Enterprise Capability Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54466 *Contributors*: Afaisandier, Asquires, Bkcase, Blawson, Dhenry, Janthony, Jgercken, Jmartin, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Systems of Systems (SoS) *Source*: https://www.sebokwiki.org/d/index.php?oldid=54467 *Contributors*: Apyster, Bkcase, Blawson, Cwright, Dhenry, Dholwell, Dnewbern, Henshaw, Janthony, Jdahmann, Jgercken, Kguillemette, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Architecting Approaches for Systems of Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54186 *Contributors*: Apyster, Bkcase, Blawson, Cwright, Dhenry, Dholwell, Dnewbern, Janthony, Jdahmann, Jgercken, Kguillemette, Skmackin, Wikiexpert, Zamoses

Socio-Technical Features of Systems of Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54469 *Contributors*: Afaisandier, Bkcase, Blawson, Cwright, Dhenry, Dholwell, Dnewbern, Hdavidz, Janthony, Jdahmann, Jgercken, Kguillemette, Mhenshaw, Radcock, Skmackin, Wikiexpert, Zamoses

Capability Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54471 *Contributors*: Apyster, Bkcase, Blawson, Cwright, Dhenry, Dholwell, Dnewbern, Janthony, Jdahmann, Jgercken, Kguillemette, Skmackin, Wikiexpert, Zamoses

Healthcare Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54474 *Contributors*: Bkcase, Nicole.hutchison, Radcock

Overview of the Healthcare Sector *Source*: https://www.sebokwiki.org/d/index.php?oldid=54475 *Contributors*: Bkcase, Nicole.hutchison, Radcock

Systems Engineering in Healthcare Delivery *Source*: https://www.sebokwiki.org/d/index.php?oldid=54476 *Contributors*: Bkcase, Nicole.hutchison

Systems Biology *Source*: https://www.sebokwiki.org/d/index.php?oldid=54477 *Contributors*: Bkcase, Nicole.hutchison

Lean in Healthcare *Source*: https://www.sebokwiki.org/d/index.php?oldid=54478 *Contributors*: Bkcase, Nicole.hutchison

Enabling Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54479 *Contributors*: Apyster, Asquires, Bkcase, Blawson, Dfairley, Dhenry, Dholwell, Gparnell, Hsillitto, Janthony, Jgercken, Kguillemette, Radcock, Smenck2, Wikiexpert, Zamoses

Enabling Businesses and Enterprises *Source*: https://www.sebokwiki.org/d/index.php?oldid=54480 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Hsillitto, Jgercken, Rbeasley, Rturner, Sjackson, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Systems Engineering Organizational Strategy *Source*: https://www.sebokwiki.org/d/index.php?oldid=54481 *Contributors*: Afaisandier, Apyster, Asquires, Bkcase, Dfairley, Dhenry, Hsillitto, Jgercken, Kguillemette, Mhenshaw, Radcock, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Determining Needed Systems Engineering Capabilities in Businesses and Enterprises *Source*: https://www.sebokwiki.org/d/index.php?oldid=54234 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Groedler, Hdavidz, Hsillitto, Jgercken, Kguillemette, Mhenshaw, Radcock, Rbeasley, Rmadachy, Rturner, Sjackson, Smenck2, Wikiexpert, Ymordecai, Zamoses

Organizing Business and Enterprises to Perform Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54482 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Hsillitto, Jgercken, Kguillemette, Mhenshaw, Qwang, Rbeasley, Sjackson, Smenck2, Wikiexpert, Ymordecai, Zamoses

Assessing Systems Engineering Performance of Business and Enterprises *Source*: https://www.sebokwiki.org/d/index.php?oldid=54325 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Hsillitto, Jgercken, Mhenshaw, Rbeasley, Smenck2, Wikiexpert, Ymordecai, Zamoses

Developing Systems Engineering Capabilities within Businesses and Enterprises *Source*: https://www.sebokwiki.org/d/index.php?oldid=54241 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Hsillitto, Jgercken, Mhenshaw, Rbeasley, Smenck2, Wikiexpert, Ymordecai, Zamoses

Culture *Source*: https://www.sebokwiki.org/d/index.php?oldid=54485 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Gparnell, Hsillitto, Jgercken, Jsnoderly, Kguillemette, Mhenshaw, Rbeasley, Rturner, Sjackson, Smenck2, Wikiexpert, Zamoses

Enabling Teams *Source*: https://www.sebokwiki.org/d/index.php?oldid=54487 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Gparnell, Jgercken, Kguillemette, Smenck2, Wikiexpert, Zamoses

Team Capability *Source*: https://www.sebokwiki.org/d/index.php?oldid=54488 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Gparnell, Hdavidz, Jgercken, Kguillemette, Skmackin, Smenck2, Wikiexpert, Zamoses

Team Dynamics *Source*: https://www.sebokwiki.org/d/index.php?oldid=54489 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Janthony, Jgercken, Skmackin, Smenck2, Wikiexpert, Zamoses

Technical Leadership in Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54493 *Contributors*: Bkcase, Cnielsen, Eleach, Hdavidz, Kguillemette, Smenck2

Enabling Individuals *Source*: https://www.sebokwiki.org/d/index.php?oldid=54494 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Gparnell, Hdavidz, Janthony, Jgercken, Skmackin, Smenck2, Wikiexpert, Zamoses

Roles and Competencies *Source*: https://www.sebokwiki.org/d/index.php?oldid=54495 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Gparnell, Hdavidz, Janthony, Jgercken, Kguillemette, Mhenshaw, Radcock, Skmackin, Smenck2, Thilburn, Wikiexpert, Zamoses

Assessing Individuals *Source*: https://www.sebokwiki.org/d/index.php?oldid=54497 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Hdavidz, Janthony, Jgercken, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Developing Individuals *Source*: https://www.sebokwiki.org/d/index.php?oldid=54492 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Gparnell, Hdavidz, Janthony, Jgercken, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Zamoses

Ethical Behavior *Source*: https://www.sebokwiki.org/d/index.php?oldid=54498 *Contributors*: Apyster, Asquires, Bkcase, Ccalvano, Dfairley, Dhenry, Dholwell, Gparnell, Hdavidz, Janthony, Jgercken, Kguillemette, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Related Disciplines *Source*: https://www.sebokwiki.org/d/index.php?oldid=54499 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Kguillemette, Radcock, SYSIND11, Smenck2, Thilburn, Wikiexpert, Ymordecai, Zamoses

Systems Engineering and Software Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54502 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Mhenshaw, Radcock, Rmadachy, Skmackin, Thilburn, Wikiexpert, Ymordecai, Zamoses

Software Engineering in the Systems Engineering Life Cycle *Source*: https://www.sebokwiki.org/d/index.php?oldid=54503 *Contributors*: Asquires, Bkcase, Radcock, Thilburn

The Nature of Software *Source*: https://www.sebokwiki.org/d/index.php?oldid=54504 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Thilburn, Wikiexpert, Ymordecai, Zamoses

An Overview of the SWEBOK Guide *Source*: https://www.sebokwiki.org/d/index.php?oldid=54259 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Hdavidz, Jgercken, Kguillemette, Smenck2, Thilburn, Wikiexpert, Ymordecai, Zamoses

Key Points a Systems Engineer Needs to Know about Software Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54506 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Eleach, Jgercken, Kguillemette, Mhenshaw, Smenck2, Thilburn, Wikiexpert, Ymordecai, Zamoses

Software Engineering Features - Models, Methods, Tools, Standards, and Metrics *Source*: https://www.sebokwiki.org/d/index.php?oldid=54507 *Contributors*: Bkcase, Thilburn

Systems Engineering and Project Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54508 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Kguillemette, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

The Nature of Project Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54509 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Kguillemette, Rturner, Smenck2, Wikiexpert, Ymordecai, Zamoses

An Overview of the PMBOK® Guide *Source*: https://www.sebokwiki.org/d/index.php?oldid=54511 *Contributors*: Asquires, Bkcase, Dhenry, Dholwell, Kguillemette, Smenck2, Wikiexpert, Ymordecai

Relationships between Systems Engineering and Project Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54512 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Dnewbern, Jgercken, Kguillemette, Mhenshaw, Rturner, Smenck2, Wikiexpert, Zamoses

The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships *Source*: https://www.sebokwiki.org/d/index.php?oldid=54513 *Contributors*: Asquires, Bkcase, Dhenry, Dholwell, Dnewbern, Kguillemette, Mhenshaw, Radcock, Smenck2, Wikiexpert

Procurement and Acquisition *Source*: https://www.sebokwiki.org/d/index.php?oldid=54514 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Jgercken, Kguillemette, Mhenshaw, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Systems Engineering and Industrial Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54516 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Dnewbern, Hsillitto, Kguillemette, SYSIND11, Smenck2, Wikiexpert

Systems Engineering and Specialty Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54517 *Contributors*: Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Nicole.hutchison, Phisterp, Radcock, Rturner, S.Jackson, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Reliability, Availability, and Maintainability *Source*: https://www.sebokwiki.org/d/index.php?oldid=54519 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Mhenshaw, Phisterp, Rturner, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Human Systems Integration *Source*: https://www.sebokwiki.org/d/index.php?oldid=54520 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Mhenshaw, Phisterp, Radcock, Sbooth, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Safety Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54521 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Jgercken, Mhenshaw, Phisterp, Radcock, Skmackin, Smenck2, Wikiexpert, Ymordecai, Zamoses

Security Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54522 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Gparnell, Jgercken, Phisterp, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Electromagnetic Interference/Electromagnetic Compatibility *Source*: https://www.sebokwiki.org/d/index.php?oldid=54523 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Gparnell, Jgercken, Jsnoderly, Mhenshaw, Nicole.hutchison, Phisterp, Radcock, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

System Resilience *Source*: https://www.sebokwiki.org/d/index.php?oldid=54524 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Gparnell, Jbrtis, Jgercken, Kguillemette, Mhenshaw, Nicole.hutchison, Phisterp, Radcock, Rturner, S.Jackson, Sjackson, Skmackin, Smenck2, Wikiexpert, Zamoses

Manufacturability and Producibility *Source*: https://www.sebokwiki.org/d/index.php?oldid=54525 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Gparnell, Jgercken, Kforsberg, Nicole.hutchison, Phisterp, Radcock, Rturner, Skmackin, Wikiexpert, Zamoses

Affordability *Source*: https://www.sebokwiki.org/d/index.php?oldid=54205 *Contributors*: Bkcase, Dhenry, Dholwell, Gparnell, Kguillemette, Phisterp, Radcock, Rmadachy, Smenck2, Wikiexpert

Environmental Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=54527 *Contributors*: Bkcase, Cnielsen, Dhenry, Dholwell, Dnewbern, Kguillemette, Mhenshaw, Phisterp, Radcock, Smenck2, Wikiexpert

Systems Engineering Implementation Examples *Source*: https://www.sebokwiki.org/d/index.php?oldid=54528 *Contributors*: Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Kguillemette, Nicole.hutchison, Radcock, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

Matrix of Implementation Examples *Source*: https://www.sebokwiki.org/d/index.php?oldid=54529 *Contributors*: Alee, Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jbrackett, Jgercken, Kguillemette, Mhenshaw, Nicole.hutchison, Radcock, Rturner, Skmackin, Smenck2, Thilburn, Wikiexpert, Zamoses

Implementation Examples *Source*: https://www.sebokwiki.org/d/index.php?oldid=54530 *Contributors*: Bkcase, Nicole.hutchison, Radcock

Complex Adaptive Taxi Service Scheduler *Source*: https://www.sebokwiki.org/d/index.php?oldid=54309 *Contributors*: Bkcase, Nicole.hutchison, Radcock

Denver Airport Baggage Handling System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54446 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Kguillemette, Nicole.hutchison, Qwang, Radcock, Skmackin, Smenck2, Thilburn, Wikiexpert, Zamoses

Global Positioning System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54533 *Contributors*: Apyster, Asquires, Bkcase, Bwhite, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Radcock, Rfreeman, Smenck2, Thilburn, Wikiexpert, Zamoses

Global Positioning System II *Source*: https://www.sebokwiki.org/d/index.php?oldid=54534 *Contributors*: Bkcase, Bwhite, Nicole.hutchison, Radcock

Next Generation Medical Infusion Pump *Source*: https://www.sebokwiki.org/d/index.php?oldid=54537 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Smenck2, Wikiexpert

Medical Radiation *Source*: https://www.sebokwiki.org/d/index.php?oldid=54536 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Hdavidz, Jbrackett, Jgercken, Kguillemette, Nicole.hutchison, Smenck2, Wikiexpert, Zamoses

Project Management for a Complex Adaptive Operating System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54538 *Contributors*: Bkcase, Nicole.hutchison, Radcock

Russian Space Agency Project Management Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54539 *Contributors*: Bkcase, Nicole.hutchison, Radcock

Successful Business Transformation within a Russian Information Technology Company *Source*: https://www.sebokwiki.org/d/index.php?oldid=54540 *Contributors*: Bkcase, Bwhite, Cnielsen, Kguillemette, Nicole.hutchison, Radcock

Federal Bureau of Investigation (FBI) Virtual Case File System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54542 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Hdavidz, Jbrackett, Jgercken, Nicole.hutchison, Skmackin, Smenck2, Wikiexpert, Zamoses

Design for Maintainability *Source*: https://www.sebokwiki.org/d/index.php?oldid=54231 *Contributors*: Bkcase, Nicole.hutchison, Radcock

Federal Aviation Administration (FAA) Advanced Automation System (AAS) *Source*: https://www.sebokwiki.org/d/index.php?oldid=54545 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Radcock, Rturner, Thilburn, Wikiexpert

Federal Aviation Administration (FAA) Next Generation Air Transportation System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54168 *Contributors*: Bkcase, Bwhite, Cnielsen, Kguillemette, Nicole.hutchison, Radcock

How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn *Source*: https://www.sebokwiki.org/d/index.php?oldid=54547 *Contributors*: Bkcase, Bwhite, Kguillemette, Nicole.hutchison, Radcock

Hubble Space Telescope *Source*: https://www.sebokwiki.org/d/index.php?oldid=54548 *Contributors*: Apyster, Asquires, Bkcase, Bwhite, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Radcock, Rfreeman, Smenck2, Wikiexpert, Zamoses

Northwest Hydro System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54549 *Contributors*: Bkcase, Nicole.hutchison

Singapore Water Management *Source*: https://www.sebokwiki.org/d/index.php?oldid=54550 *Contributors*: Alee, Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Radcock, Rturner, Skmackin, Wikiexpert, Zamoses

Submarine Warfare Federated Tactical Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=54551 *Contributors*: Bkcase, Nicole.hutchison

UK West Coast Route Modernisation Project *Source*: https://www.sebokwiki.org/d/index.php?oldid=54552 *Contributors*: Alee, Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Radcock, Rturner, Skmackin, Smenck2, Thilburn, Wikiexpert, Zamoses

Virginia Class Submarine *Source*: https://www.sebokwiki.org/d/index.php?oldid=54553 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Radcock, Skmackin, Smenck2, Wikiexpert, Zamoses

Applying a Model-Based Approach to Support Requirements Analysis on the Thirty-Meter Telescope *Source*: https://www.sebokwiki.org/d/index.php?oldid=54250 *Contributors*: Bkcase, Nicole.hutchison

Miniature Seeker Technology Integration Spacecraft *Source*: https://www.sebokwiki.org/d/index.php?oldid=54555 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Nicole.hutchison, Skmackin, Smenck2, Wikiexpert, Zamoses

Standard Korean Light Transit System *Source*: https://www.sebokwiki.org/d/index.php?oldid=54556 *Contributors*: Apyster, Asquires, Bkcase, Ccalvano, Dhenry, Dholwell, Hdavidz, Jgercken, Mhenshaw, Nicole.hutchison, Radcock, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

Image Sources, Licenses and Contributors

File:Rob_Sm for Web.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Rob_Sm_for_Web.jpg *License*: unknown *Contributors*: -

File:RobSignature2.jpeg *Source*: https://www.sebokwiki.org/d/index.php?title=File:RobSignature2.jpeg *License*: unknown *Contributors*: -

File:SEBoK Navigation Introduction.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Introduction.PNG *License*: unknown *Contributors*: Bkcase

File:SEBoK Navigation Structure.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Structure.PNG *License*: unknown *Contributors*: Bkcase

File:SE_Key_Concepts.jpeg *Source*: https://www.sebokwiki.org/d/index.php?title=File:SE_Key_Concepts.jpeg *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:Scope_BoundariesSE_PM_SM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Scope_BoundariesSE_PM_SM.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:P1_Scope_and_Con_SE_and_Eng_Sys_Proj_LF_BB.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:P1_Scope_and_Con_SE_and_Eng_Sys_Proj_LF_BB.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:NASA Image Part 1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:NASA_Image_Part_1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:P1_EconValueSE_RiskBalancedfig2_BB.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:P1_EconValueSE_RiskBalancedfig2_BB.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SE Core Concept Relationships.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SE_Core_Concept_Relationships.PNG *License*: unknown *Contributors*: Bkcase

File:SE Core Concept Relationships Example.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SE_Core_Concept_Relationships_Example.PNG *License*: unknown *Contributors*: Bkcase

File:Core SEBoK Concepts.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Core_SEBoK_Concepts.PNG *License*: unknown *Contributors*: Bkcase

File:SEBoK Navigation Foundation.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Foundation.PNG *License*: unknown *Contributors*: Bkcase

File:IFSR_SPF_August_2013.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:IFSR_SPF_August_2013.jpg *License*: unknown *Contributors*: Smenck2

File:IFSR_ISA_July_2012_REV.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:IFSR_ISA_July_2012_REV.png *License*: unknown *Contributors*: Bkcase, Smenck2

File:Fig_1_System_Fundamentals_and_Engineered_Systems_RA.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_System_Fundamentals_and_Engineered_Systems_RA.png *License*: unknown *Contributors*: Bkcase, Mhenshaw

File:Fig_1_Systems_Science_and_Systems_Thinking_RA.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_Systems_Science_and_Systems_Thinking_RA.png *License*: unknown *Contributors*: Bkcase, Mhenshaw

File:Fig2_Systems_Thinking_and_Systems_Science_RA.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig2_Systems_Thinking_and_Systems_Science_RA.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:Picture1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Picture1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:060611_SF_System_Conept_Model-Top_Levelnofig10.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:060611_SF_System_Conept_Model-Top_Levelnofig10.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Fig_1_Systems_Engineering_and_the_Systems_Approach_RA.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_Systems_Engineering_and_the_Systems_Approach_RA.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:052311_SJ_System_Coupling_Diagram.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:052311_SJ_System_Coupling_Diagram.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Ellipse Graphic J Ring.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Ellipse_Graphic_J_Ring.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:Fig_3_Complex_Systems_Engineering_Diagram_Sillitto2010.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_3_Complex_Systems_Engineering_Diagram_Sillitto2010.png *License*: unknown *Contributors*: Bkcase, Mhenshaw

File:Fig_1_SA_Activities_Through_Life_RA.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_SA_Activities_Through_Life_RA.png *License*: unknown *Contributors*: Bkcase, Cnielsen, Mhenshaw

File:SEBoK Navigation Management.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Management.PNG *License*: unknown *Contributors*: Bkcase

File:Mapping_of_tech_topics_SEBoK_with_ISO_IEC_15288techPro_060612.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Mapping_of_tech_topics_SEBoK_with_ISO_IEC_15288techPro_060612.jpg *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:062211_BL_Paradigm.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:062211_BL_Paradigm.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:Fig_1_A_generic_life_cycle_KF.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_A_generic_life_cycle_KF.png *License*: unknown *Contributors*: Bkcase

File:SE Hump diagram.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SE_Hump_diagram.PNG *License*: unknown *Contributors*: Bkcase

File:Ex_Itera_of_processes_related_to_Sys_Def_AF_052312.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Ex_Itera_of_processes_related_to_Sys_Def_AF_052312.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:JS_Figure_1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Hierarchical_decomposition_of_a_system-of-interest_060612.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Hierarchical_decomposition_of_a_system-of-interest_060612.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Recursion_of_processes_on_layers_060612.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Recursion_of_processes_on_layers_060612.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Needs-to-requirements-ryan.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Needs-to-requirements-ryan.PNG *License*: unknown *Contributors*: Bkcase

File:Fig 1 Primary models of incremental and evolutionary development KF.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_Primary_models_of_incremental_and_evolutionary_development_KF.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:KF_EvolutionaryConcurrentChange.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_EvolutionaryConcurrentChange.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_VeeModel_Left.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_VeeModel_Left.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Fig_2_Life_Cycle_Stages_Vee_KF.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_2_Life_Cycle_Stages_Vee_KF.png *License*: unknown *Contributors*: Bkcase

File:JS_Figure_2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_GenericStageStructure.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_GenericStageStructure.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Comparisons_of_life_cycle_models.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Comparisons_of_life_cycle_models.PNG *License*: unknown *Contributors*: Janthony, Mhenshaw

File:KF_SchedulingDevelopment.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_SchedulingDevelopment.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_ILSSystemLifeCycle.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_ILSSystemLifeCycle.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_VeeModel_Right.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_VeeModel_Right.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_IncrementalDevelopment_Multiple.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_IncrementalDevelopment_Multiple.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Incremental_Development_with_a_single_delivery.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Incremental_Development_with_a_single_delivery.PNG *License*: unknown *Contributors*: Janthony, Mhenshaw

File:Evolutionary_Generic_Model.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Evolutionary_Generic_Model.PNG *License*: unknown *Contributors*: Janthony, Mhenshaw

File:KF_EvolutionComponents_Orbiter.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_EvolutionComponents_Orbiter.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_IncrementalBuildCycles.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_IncrementalBuildCycles.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_IncrementalCommitmentSpiral.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_IncrementalCommitmentSpiral.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_Phase_GenericIncremental.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_Phase_GenericIncremental.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_ICSMActivityCategories.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_ICSMActivityCategories.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:KF_FeasibilityEvidenceDescription.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_FeasibilityEvidenceDescription.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Tale_of_Two_Implementations_Schwaber.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Tale_of_Two_Implementations_Schwaber.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Example_of_Architected_Agile_Process_Replacement_070912.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Example_of_Architected_Agile_Process_Replacement_070912.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Generic_(T)_Stage_Structure_of_System_Life_Cycle_Models_(Lawson_2010,_Figure_6-2).png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Generic_(T)_Stage_Structure_of_System_Life_Cycle_Models_(Lawson_2010,_Figure_6-2).png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:@@BKCASE_Wiki_Section_2.5_Fig_2a_PDF_110820.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Wiki_Section_2.5_Fig_2a_PDF_110820.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:@@BKCASE_Wiki_Section_2.5_Fig_2b_PDF_110820.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Wiki_Section_2.5_Fig_2b_PDF_110820.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Figure_3._Spiral_Model_support_for_Process_Models,_Product_Models....png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Figure_3._Spiral_Model_support_for_Process_Models,_Product_Models....png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:T-Model_for_Software_System_(Lawson_2010,_Figure_6-3).png *Source*: https://www.sebokwiki.org/d/index.php?title=File:T-Model_for_Software_System_(Lawson_2010,_Figure_6-3).png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Iteration_through_Life_Cycle_Stages_(Lawson_2010,_Figure_6-4).png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Iteration_through_Life_Cycle_Stages_(Lawson_2010,_Figure_6-4).png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:@@BKCASE_Sect_2.5_Fig_6A.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Sect_2.5_Fig_6A.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:@@BKCASE_Sect_2.5_Fig_6B.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Sect_2.5_Fig_6B.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:QQBKCASE_Sect_2.5_Fig_6C.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:QQBKCASE_Sect_2.5_Fig_6C.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Enterprise_Strategy_and_Concept_Development.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Enterprise_Strategy_and_Concept_Development.PNG *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv05_KA-SystDef_Cycle_of_needs.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystDef_Cycle_of_needs.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:SEBoKv05_KA-SystDef_Top-down_development_of_design_and_requirements.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystDef_Top-down_development_of_design_and_requirements.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Recursive_Instantiation_of_Def_Process_AF_071112.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Recursive_Instantiation_of_Def_Process_AF_071112.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv075_KA-SystDef_Progressive_Approach_for_Designing.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Progressive_Approach_for_Designing.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv075_KA-SystDef_Complete_Interface_Representation.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Complete_Interface_Representation.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Decomposition_of_Functions_AF_071112(2).png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Decomposition_of_Functions_AF_071112(2).png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Illustration_of_a_scenario_(eFFBD)_AF_071112.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Illustration_of_a_scenario_(eFFBD)_AF_071112.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Illustration_of_a_scenario_Activity_Diagram_AF_071112.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Illustration_of_a_scenario_Activity_Diagram_AF_071112.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv075_KA-SystDef_Scenario_of_Operational_Modes.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Scenario_of_Operational_Modes.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv05 KA-SystDef Temporal and decision hierarchy levels.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystDef_Temporal_and_decision_hierarchy_levels.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv075_KA-SystDef_Limited_nb_in_decomposition.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Limited_nb_in_decomposition.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv075_KA-SystDef_Encapsulation.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Encapsulation.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:JS_Figure_3.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_3.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv05_KA-SystRealiz_how_outputs_of_Definition_relate_to_Implementation.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_how_outputs_of_Definition_relate_to_Implementation.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:JS_Figure_5.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_5.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv05_KA-SystRealiz_Implementation_relationships.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_Implementation_relationships.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Limits_of_integration_activities.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Limits_of_integration_activities.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv05_KA-SystRealiz_Integration_relationships.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_Integration_relationships.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:JS_Figure_9.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_9.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Definition_and_usage_of_a_Verification_Action.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Definition_and_usage_of_a_Verification_Action.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Definition_and_usage_of_a_Validation_Action.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Definition_and_usage_of_a_Validation_Action.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Verification_and_Validation_level_per_level.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Verification_and_Validation_level_per_level.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv05_KA-SystRealiz_V&V_Actions_upper_levels.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_V&V_Actions_upper_levels.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoKv05_KA-SystRealiz_V&V_Actions_lower_levels.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_V&V_Actions_lower_levels.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:052611_SJ_notional_reliability_analysis.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:052611_SJ_notional_reliability_analysis.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Affordable_Sys_Ops_Effect_DAU_GB_Roedler.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Affordable_Sys_Ops_Effect_DAU_GB_Roedler.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Sustainment_Implementation_Illustration_Logistics_Roedler.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Sustainment_Implementation_Illustration_Logistics_Roedler.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:semp_and_integrated_plans.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Semp_and_integrated_plans.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Measurement_Process_Model-Figure_1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Measurement_Process_Model-Figure_1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Composition_of_Leading_Indicator-Figure_2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Composition_of_Leading_Indicator-Figure_2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Technical_Measures_Relationship-Figure_3.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Technical_Measures_Relationship-Figure_3.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Decision_Mgt_Process_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Decision_Mgt_Process_DM.png *License*: unknown *Contributors*: Smenck2

File:Fund_Obj_Hierarchy_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fund_Obj_Hierarchy_DM.png *License*: unknown *Contributors*: Smenck2

File:Value_Function_Example_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Value_Function_Example_DM.png *License*: unknown *Contributors*: Smenck2

File:Swing_Weight_Matrix_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Swing_Weight_Matrix_DM.png *License*: unknown *Contributors*: Smenck2

File:Descript_of_Alt_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Descript_of_Alt_DM.png *License*: unknown *Contributors*: Smenck2

File:ALT_Scores_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ALT_Scores_DM.png *License*: unknown *Contributors*: Smenck2

File:Value_Scorecard_w_Heat_Map_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Value_Scorecard_w_Heat_Map_DM.png *License*: unknown *Contributors*: Smenck2

File:Eq_1.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Eq_1.jpg *License*: unknown *Contributors*: Smenck2

File:Eq_2.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Eq_2.jpg *License*: unknown *Contributors*: Smenck2

File:Value_Comp_Graph_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Value_Comp_Graph_DM.png *License*: unknown *Contributors*: Smenck2

File:Ex_Stakeholder_Value_Scat_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Ex_Stakeholder_Value_Scat_DM.png *License*: unknown *Contributors*: Smenck2

File:Uncertainty_on_Perf_Value_from_Monte_DM.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Uncertainty_on_Perf_Value_from_Monte_DM.png *License*: unknown *Contributors*: Smenck2

File:Cm_functions.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Cm_functions.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:Cm_change_control_process.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Cm_change_control_process.png *License*: unknown *Contributors*: Bkcase, Mhenshaw, Smenck2

File:InfoMgtProcess_Fig1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:InfoMgtProcess_Fig1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ArchitectureDesignConsid_Fig2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ArchitectureDesignConsid_Fig2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:052411_BSBW_The_Vee_Model.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:052411_BSBW_The_Vee_Model.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Breadth_and_Depth_of_Key_SE_Related_Standards.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Breadth_and_Depth_of_Key_SE_Related_Standards.PNG *License*: unknown *Contributors*: Bkcase, Janthony, Mhenshaw

File:Standards_Alignment_Results_as_of_May_2011.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Standards_Alignment_Results_as_of_May_2011.PNG *License*: unknown *Contributors*: Bkcase, Janthony, Mhenshaw

File:Fig_3_Growing_Industry_Collaboration_GR.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_3_Growing_Industry_Collaboration_GR.png *License*: unknown *Contributors*: Bkcase, Mhenshaw

File:Standards_Alignment_Results_as_of_May_2011a.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Standards_Alignment_Results_as_of_May_2011a.PNG *License*: unknown *Contributors*: Janthony, Mhenshaw

File:Potential_Standards_Influence_of_Organization_and_Project_Processes.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Potential_Standards_Influence_of_Organization_and_Project_Processes.PNG *License*: unknown *Contributors*: Janthony, Mhenshaw

File:SEBoK Navigation Applications.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Applications.PNG *License*: unknown *Contributors*: Bkcase

File:PSE_Intro_Fig1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_Intro_Fig1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_Intro_Fig2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_Intro_Fig2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_Intro_Fig3.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_Intro_Fig3.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_Intro_Fig4.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_Intro_Fig4.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:IPDP PSE Background Figure 1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:IPDP_PSE_Background_Figure_1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PSEB_Fig2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PSEB_Fig2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PAAS_Fig1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PAAS_Fig1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PAAS_Fig2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PAAS_Fig2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PSEKA_Fig1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PSEKA_Fig1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PSEKA_Fig2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PSEKA_Fig2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PSEKA_Fig3.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PSEKA_Fig3.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PSEKA_Fig4.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PSEKA_Fig4.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PSE_PSESA_Fig1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:PSE_PSESA_Fig1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SSE_SSB_Fig1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SSE_SSB_Fig1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SSE_FOS_Fig1_no_white_space.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SSE_FOS_Fig1_no_white_space.png *License*: unknown *Contributors*: Smenck2

File:SSE_FOS_Fig3.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SSE_FOS_Fig3.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SSE_FOS_Fig4.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SSE_FOS_Fig4.PNG *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SSE_VoSSE_Fig1v2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:SSE_VoSSE_Fig1v2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F01.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F01.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F02.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F02.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F03.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F03.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F04.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F04.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F05.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F05.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F07.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F07.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F08.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F08.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F09.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F09.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F10.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F10.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F11.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F11.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F12.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F12.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F13.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F13.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F15.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F15.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F06.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F06.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F30_Process_Breakdown.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F30_Process_Breakdown.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F21.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F21.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ESE-F22.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ESE-F22.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Use Cases ISO 14971 15288.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Use_Cases_ISO_14971_15288.png *License*: unknown *Contributors*: Bkcase

File:SEBoK Navigation Enable.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Enable.PNG *License*: unknown *Contributors*: Bkcase

File:Organizational_coupling_diagram_v2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Organizational_coupling_diagram_v2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:System_enterprises_and_organizations_v2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:System_enterprises_and_organizations_v2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Culture_competence_team_performance_and_individual_competence.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Culture_competence_team_performance_and_individual_competence.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Part_5_(Organization)_Concept_maps_additions_hgs_15_Aug.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Part_5_(Organization)_Concept_maps_additions_hgs_15_Aug.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Picture1_HGS.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Picture1_HGS.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:TPM_Chart_from_INCOSE_SELIG.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:TPM_Chart_from_INCOSE_SELIG.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Concept_map_for_businesses_and_enterprises_topics.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Concept_map_for_businesses_and_enterprises_topics.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Fairley_Fig_1_(2)_Layer_1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fairley_Fig_1_(2)_Layer_1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Fairley_Fig_2_Layer_1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fairley_Fig_2_Layer_1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Dimensions_of_Communication_Styles.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Dimensions_of_Communication_Styles.png *License*: unknown *Contributors*: Smenck2

File:Layered_and_Multi-dimensional_in_the_Engineering_Layer.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Layered_and_Multi-dimensional_in_the_Engineering_Layer.PNG *License*: unknown *Contributors*: Janthony, Mhenshaw, Smenck2

File:MITRE_Enterprise_Systems_Engineering_Framework.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:MITRE_Enterprise_Systems_Engineering_Framework.PNG *License*: unknown *Contributors*: Janthony, Mhenshaw, Smenck2

File:SEBoK Navigation Related.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Related.PNG *License*: unknown *Contributors*: Bkcase

File:Aligned-Process-Models.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Aligned-Process-Models.PNG *License*: unknown *Contributors*: Bkcase

File:PM-SE1.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:PM-SE1.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:PM-SE2.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:PM-SE2.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:P6_Fig1_The_Organizational_Continuum_KN.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:P6_Fig1_The_Organizational_Continuum_KN.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ACQProcessModel_NoWhiteS.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ACQProcessModel_NoWhiteS.png *License*: unknown *Contributors*: Dhenry, Mhenshaw, Smenck2

File:RelatingACQtoRFP_NoWhiteS.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:RelatingACQtoRFP_NoWhiteS.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Fig._1_Integration_Process_for_Specialty_Engineering.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig._1_Integration_Process_for_Specialty_Engineering.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Fault_tree.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fault_tree.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Simple_RBD.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Simple_RBD.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Figure 1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Figure_1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Figure 2.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Figure_2.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:Disruption_Diagram.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Disruption_Diagram.PNG *License*: unknown *Contributors*: Bkcase, Mhenshaw

File:StateTransitionModel.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:StateTransitionModel.png *License*: unknown *Contributors*: Nicole.hutchison

File:Environmental_Engineering_HighRes.jpg *Source*: https://www.sebokwiki.org/d/index.php?title=File:Environmental_Engineering_HighRes.jpg *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:SEBoK Navigation Examples.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Examples.PNG *License*: unknown *Contributors*: Bkcase

File:Casestudies-2x2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Casestudies-2x2.png *License*: unknown *Contributors*: Bkcase

File:Adaptive-scheduling-taxi.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Adaptive-scheduling-taxi.PNG *License*: unknown *Contributors*: Bkcase

File:Enterprise-ontology.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Enterprise-ontology.PNG *License*: unknown *Contributors*: Bkcase

File:Resources-allocation.PNG *Source*: https://www.sebokwiki.org/d/index.php?title=File:Resources-allocation.PNG *License*: unknown *Contributors*: Bkcase

File:MissionAndCapabilities.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:MissionAndCapabilities.png *License*: unknown *Contributors*: Bkcase

File:V Model Process of Transformation.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:V_Model_Process_of_Transformation.png *License*: unknown *Contributors*: Bkcase

File:Results of Tranformation.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:Results_of_Tranformation.png *License*: unknown *Contributors*: Bkcase

File:AN BSY-1 Submarine Combat System.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:AN_BSY-1_Submarine_Combat_System.png *License*: unknown *Contributors*: Bkcase

File:VA combat system architecture.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:VA_combat_system_architecture.png *License*: unknown *Contributors*: Bkcase

File:ThirtyMeterTelescope_Figure1_TelescopeImage.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ThirtyMeterTelescope_Figure1_TelescopeImage.png *License*: unknown *Contributors*: Nicole.hutchison

File:ThirtyMeterTelescope_Figure2_EarlyLightLocations.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ThirtyMeterTelescope_Figure2_EarlyLightLocations.png *License*: unknown *Contributors*: Nicole.hutchison

File:ThirtyMeterTelescope_Figure3.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ThirtyMeterTelescope_Figure3.png *License*: unknown *Contributors*: Nicole.hutchison

File:ThirtyMeterTelescope_Figure4.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ThirtyMeterTelescope_Figure4.png *License*: unknown *Contributors*: Nicole.hutchison

File:ThirtyMeterTelescope_Figure5.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ThirtyMeterTelescope_Figure5.png *License*: unknown *Contributors*: Nicole.hutchison

File:ThirtyMeterTelescope Figure6.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ThirtyMeterTelescope_Figure6.png *License*: unknown *Contributors*: Nicole.hutchison

File:ThirtyMeterTelescope_Figure7.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ThirtyMeterTelescope_Figure7.png *License*: unknown *Contributors*: Nicole.hutchison

File:ChoeKimFigure1.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ChoeKimFigure1.png *License*: unknown *Contributors*: Mhenshaw, Smenck2

File:ChoeKimFigure2.png *Source*: https://www.sebokwiki.org/d/index.php?title=File:ChoeKimFigure2.png *License*: unknown *Contributors*: Mhenshaw, Smenck2